U.S. DEPARTMENT OF ENERGY

DRAFT

Prepared for the U.S. Department of Energy
Under Contract DE-AC05-76RL01830

# Developing a Value Added Process (VAP) using ARM's Interactive Software Development Environment (ISDE)

Revision 0

KL Gaustad

June 2011

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

PACIFIC NORTHWEST NATIONAL LABORATORY
*operated by*
BATTELLE
*for the*
UNITED STATES DEPARTMENT OF ENERGY
*under Contract DE-AC05-76RL01830*

# Developing a Value Added Process (VAP) using ARM's Interactive Software Development Environment (ISDE)

KL Gaustad

May 2011

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory

Richland, Washington  99352

# Acronyms and Abbreviations

| | |
|---|---|
| ARM | Atmospheric Radiation Measurement |
| BOD | birth of a datastream |
| DOD | data object design |
| DSDB | Data System Database |
| ISDE | Interactive Software Development Environment |
| netCDF | Network Common Data Form |
| PCM | Process Configuration Manager |
| QC | Quality Check |
| VAP | Value Added Process |

# Contents

# Figures

**Figures**

# 1 Overview of ARM's ISDE

The Atmospheric Radiation Measurement (ARM) program's Interactive Software Development Environment (ISDE) is a suite of tools, C libraries, structures, and interfaces designed to simplify the development of ARM Value Added Process (VAP) applications to the extent that the only source code that a user must write is that of the VAP's scientific algorithm. An ARM VAP is a process that reads in netCDF data and produces netCDF files that adhere to ARM's data standards.

## 1.1 Creating an ARM VAP

Creating an ARM VAP consists of seven steps as shown in Figure 1.1:

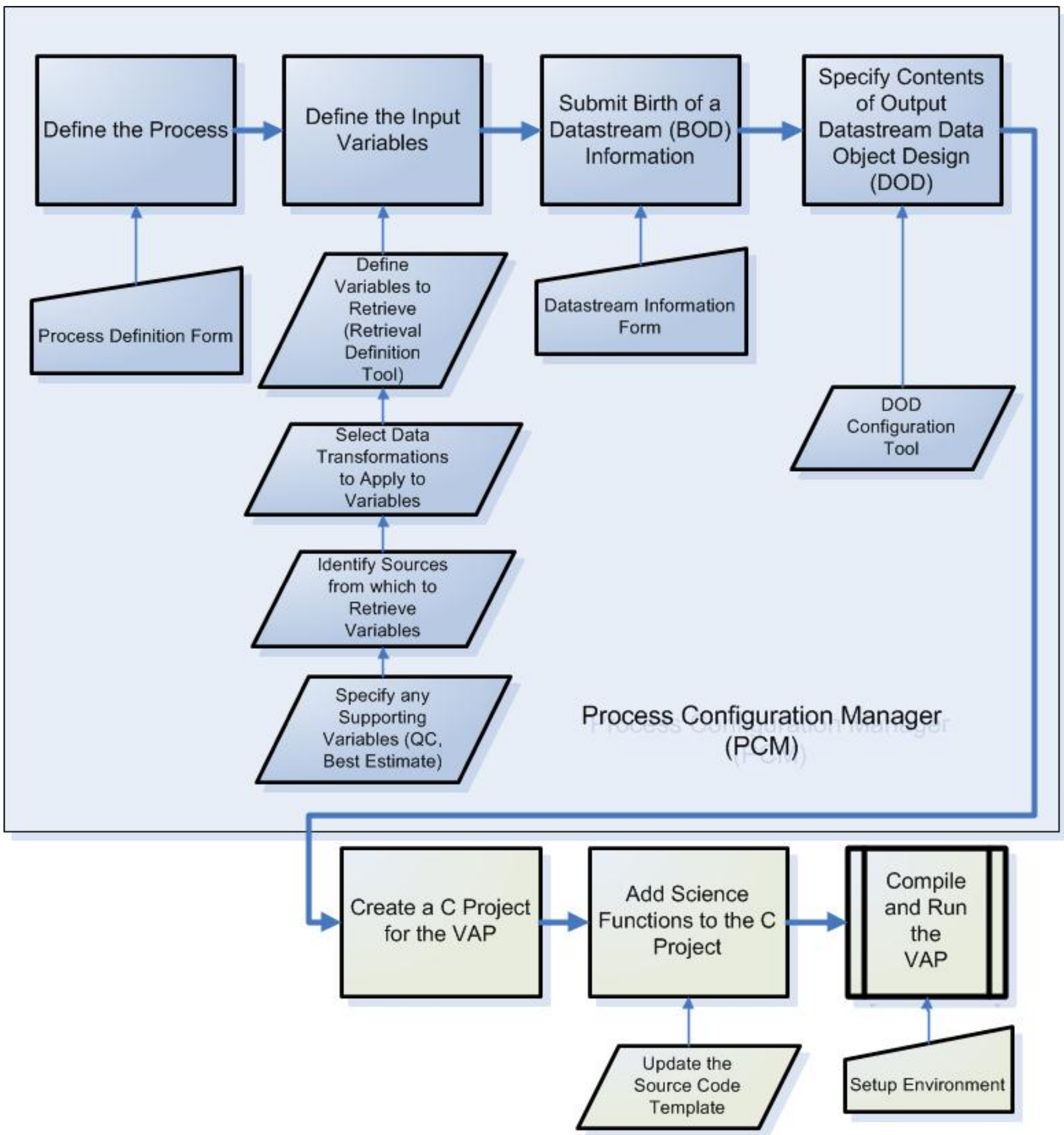


**Figure 1.1. Creating a VAP**

Currently the libraries support the basic functionality of reading data previously processed by an ARM application, performing simple transformations, and writing processed data to a netCDF file. Web based interfaces are used to gather the information associated with the first four steps and store it in the ARM Data System Database (DSDB). A code generation function, 'isdetemplater', uses the DSDB entries and template prototypes to create the VAP's C project files. The initial project produced will run "out of the box", creating output as indicated with missing values assigned to those variables whose values are to be determined as part of the scientific analysis. The VAP's input and output specifications are stored in individual header files. This allows a user to add, remove, or alter input or output variables via the web based interfaces and recreate just the affected header files with no impact to their main source code.

For each of the seven stages of VAP development this document provides a high level description of that stage of development, a description of the ISDE tools to support that stage, and a step by step guide that walks a user through the process via the implementation of a VAP named 'example_vap'. The main sections of the document and ISDE tools associated with each are:

2. Define process (Process Configuration Manager, Process Definition Form)
3. Define variables to retrieve (Retrieval Definition Tool)
    a. Define variables to retrieve
    b. Select data transformations to apply to the variables
    c. Identify source(s) from which to retrieve variables
    d. Specify supporting variables (QC, best estimate, data source, etc.) to retrieve in addition to the variable itself
4. Birth of a Datastream (BOD) documentation (Datastream Information Form)
5. Define content of Output Datastream(s) (DOD Configuration Tool)
6. Create a C Project
7. Setup environment, compile, and run.
8. Update the source code template to add science functions

# 2 Define an ARM VAP

Defining an ARM process consists of defining its inputs, outputs, and documenting where it will run via the Process Definition Tool.

## 2.1 Process Configuration Manager

The Process Configuration Manager (PCM) is a master interface from which a user can access the Process Definition Tool and other tools used to view, edit, create, and delete all aspects of a VAP process and its datastream. The currently loaded interfaces are displayed on the right panel, while access to ARM's datastreams and processes is maintained on the left via the 'Processes' and 'Datastreams' tabs. A user can have multiple tools open and be simultaneously viewing or editing datastreams, processes, and DODs. Each instance of an active tool is maintained in a set of tabs located along the top of the PCM's right panel as shown in Figure 2.1.

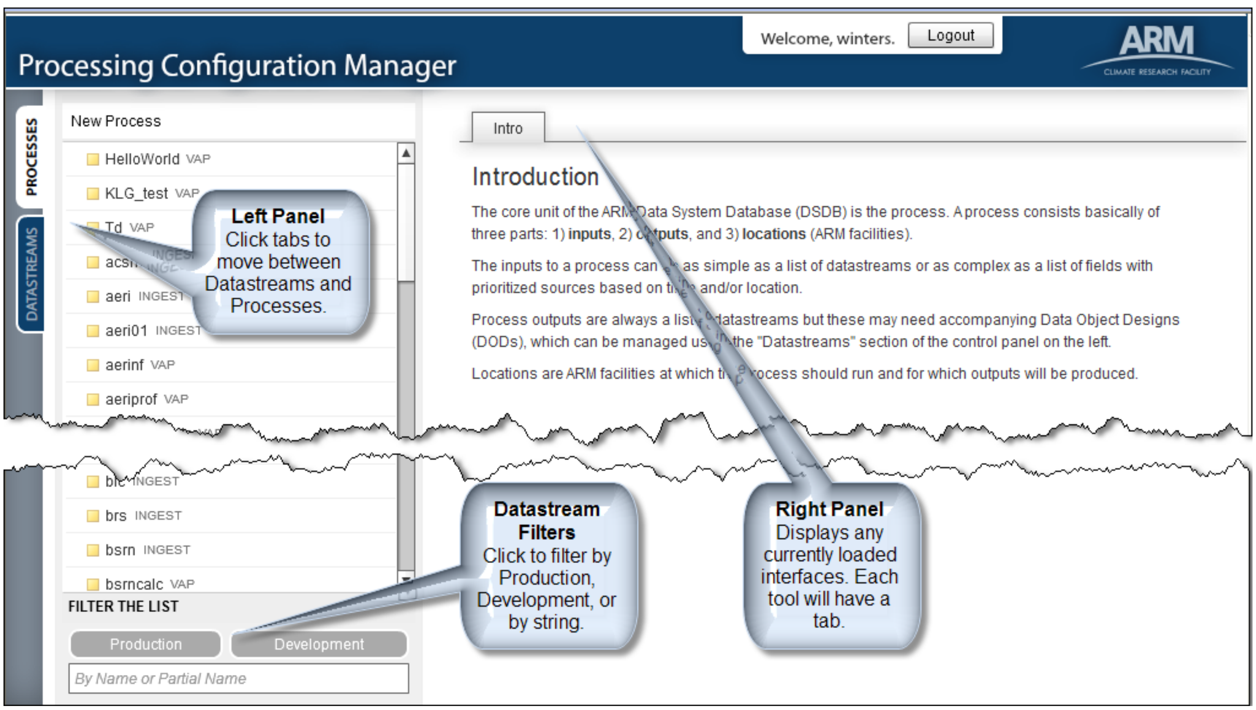


**Figure 2.1. Process Configuration Manager (PCM)**

By default the PCM will enable the Datastream tab of the left panel and display the 'Intro' tab on the right panel, as shown in Figure 2.1. Note in Figure 2.1, in the left frame, below the 'Filter the List' heading, the 'Production' and 'Development' tabs are grayed out. This indicates that the list of datstreams displayed is a combined list of both production and development datsatreams. To filter out the development datastreams, select the 'Production' button. To filter datastreams by a string, enter the string into the blank cell below the 'Production' button.

From the PCM:Datastreams page a user can view existing datastreams, their associated DODs, create new datastreams, define BODs for a datastream, and define DODs for a datastream. From the PCM:Processes page a user can view an existing process, or create a new process as shown in Figure 2.2. Processes are displayed for definition and updates via the Process Definition Tool (see Figure 2.3).

## 2.2 Process Definition

The Process Definition form consists of the following:

Process Name: Required.

- Name of the process.
  - This name must match the VAP Name entered to the Categorization element of the Datastream Information Tool.

Process Type: Required.

- Type must be selected as either an ingest or a VAP.

Facilities: Required

- The site/facility combinations at which the new process will be run.
  - Facilities are specified by selecting the applicable locations from the drop list of available values.
  - A VAP can only run at facilities that are documented in the DSDB.

Inputs: Required

- A specification of the inputs that will be needed for the process to run.
  - A VAP must use a retrieval to specify its inputs.
  - An ingest's input(s) are specified by selecting an existing datastream class and data level from the drop list of available values, or by entering a datastream base platform name (i.e. sirs, mfr10m, etc.) and datalevel (typically 00) that conform to ARM naming standards.

Outputs: Required

- List of datastream(s), defined by the datastream base platform and data level, produced by the process.
  - Specified by selecting an existing datastream class and data level from the drop list of available values, or by entering a datastream base platform name and datalevel that conform to ARM naming standards.

## 2.3 Defining a New VAP

To begin the process of defining a new VAP, perform the following steps.

1. To open the Processing Configuration Manager (PCM), login at https://engineering.arm.gov/pcm/Main.html using your ARM wiki user name and password as shown in Figure 2.2.

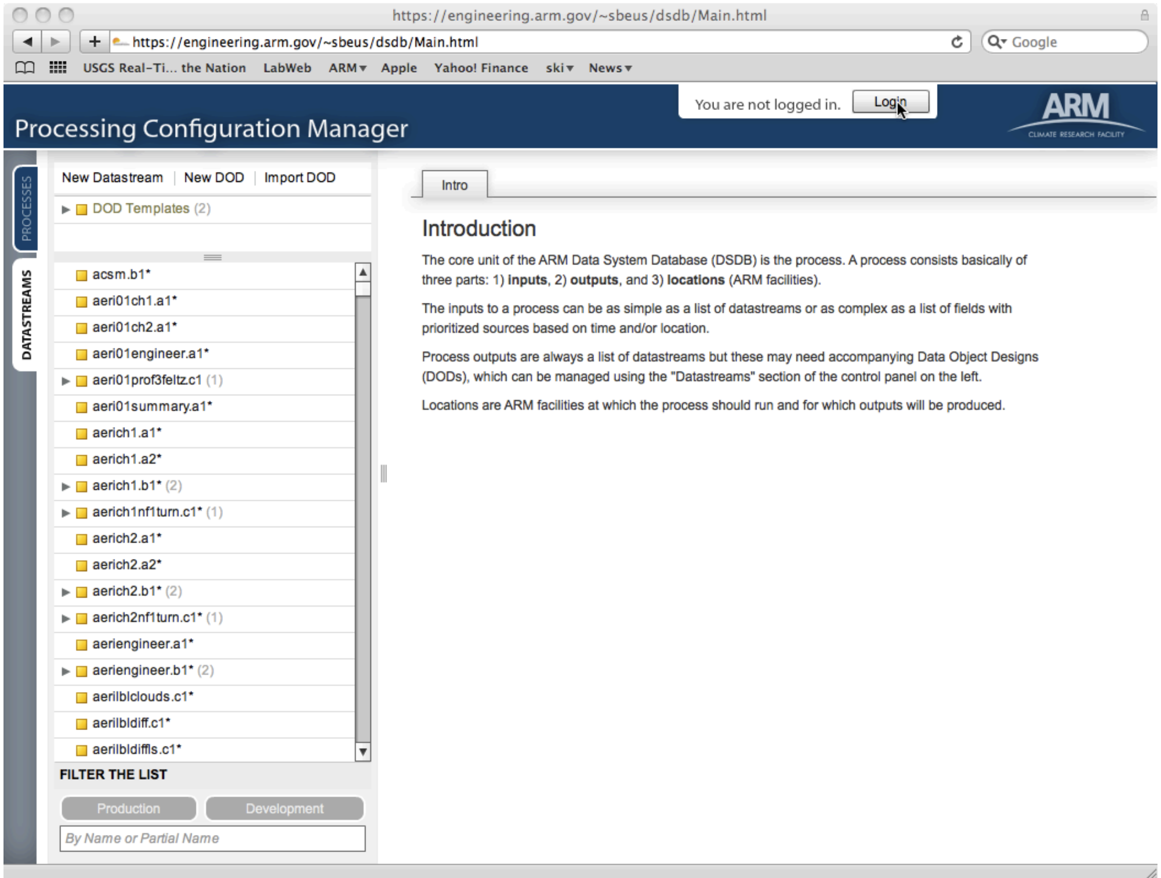


**Figure 2.2. Login from the Process Configuration Manager**

2. Select the Processes tab in the upper left panel of the PCM as shown in Figure 2.3.

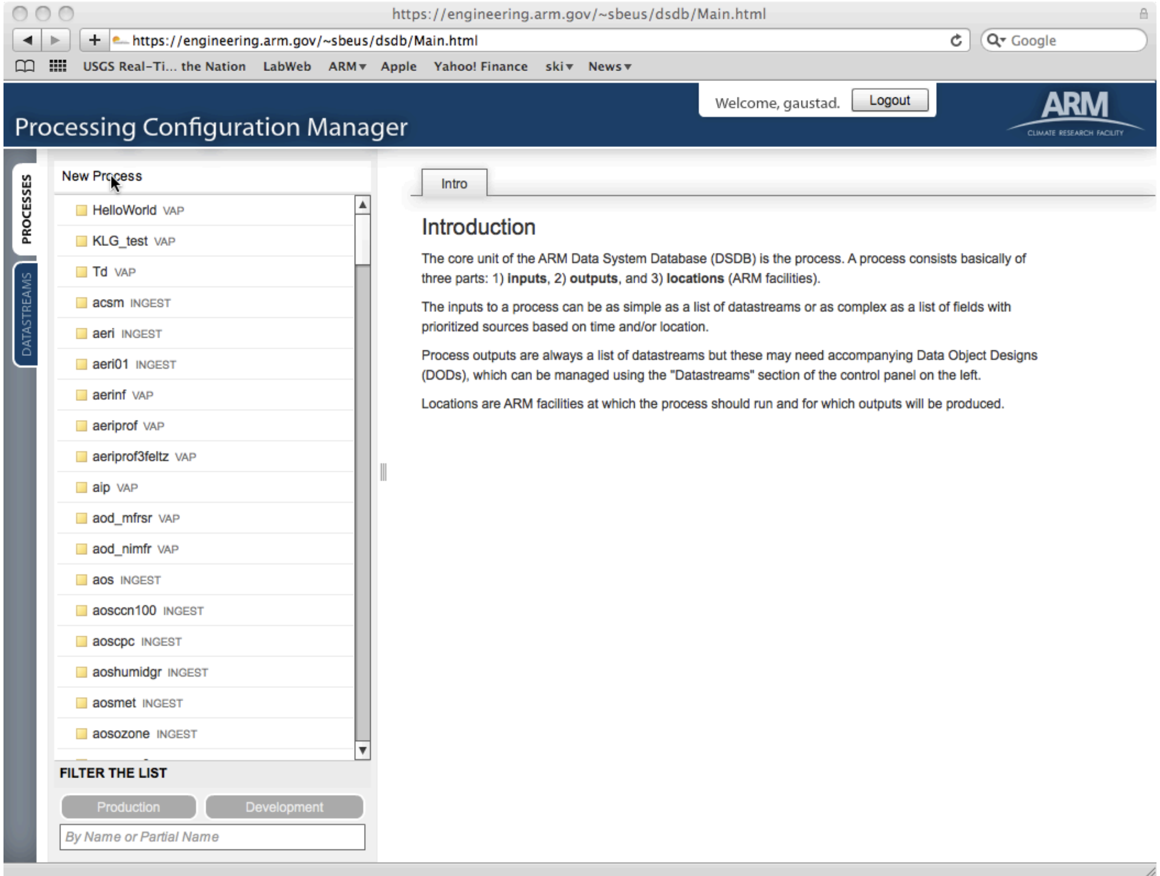


**Figure 2.3. Selecting New Process Button to bring up Process Definition Form**

3. Define a new VAP:

   a. Enter a name for the process. For this tutorial, we will name the process *example_vap*.

   b. Select the process type as VAP.

   c. Enter the facilities at which the VAP should run by selecting site and facility pairs from the provided drop list as shown in Figure 2.4. To display the drop list, start typing the name of the facility. You can then select the facility from the list of candidates. For this tutorial, select *NSA C1* to start. To delete a selection, click the X beside the item name. If a site is not listed it needs to be loaded into the database. Contact Sherman Beus for further information.

   d. Enter the output datastream name, which for this tutorial is *examplevap.c1* (note that site and facility are not included in this definition). As in step c, a drop list will display candidate datastream names.

   e. Select the 'Save' button to save the entries to the DSDB.

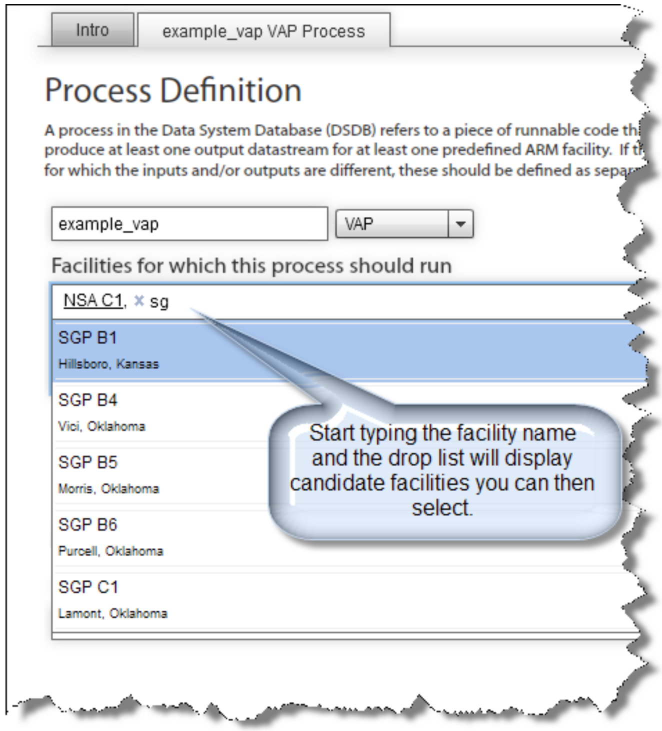


**Figure 2.4. Selecting a Facility from the Drop List**

For the example shown in Figure 2.5 the VAP process created is 'example_vap', it runs at the sgp C1, sgp B1, and nsa C1 facilities, and produces the output datastream examplevap.c1. If the examplevap.c1 datastream has not been previously defined, saving the process information to the DSDB will result in the addition of the examplevap.c1 to the list of datastreams available from the PCM:Datastreams view. Note the Process Definition form is labeled as an 'example_vap Process Form' tab at the top of the right panel next to the 'Intro' tab.

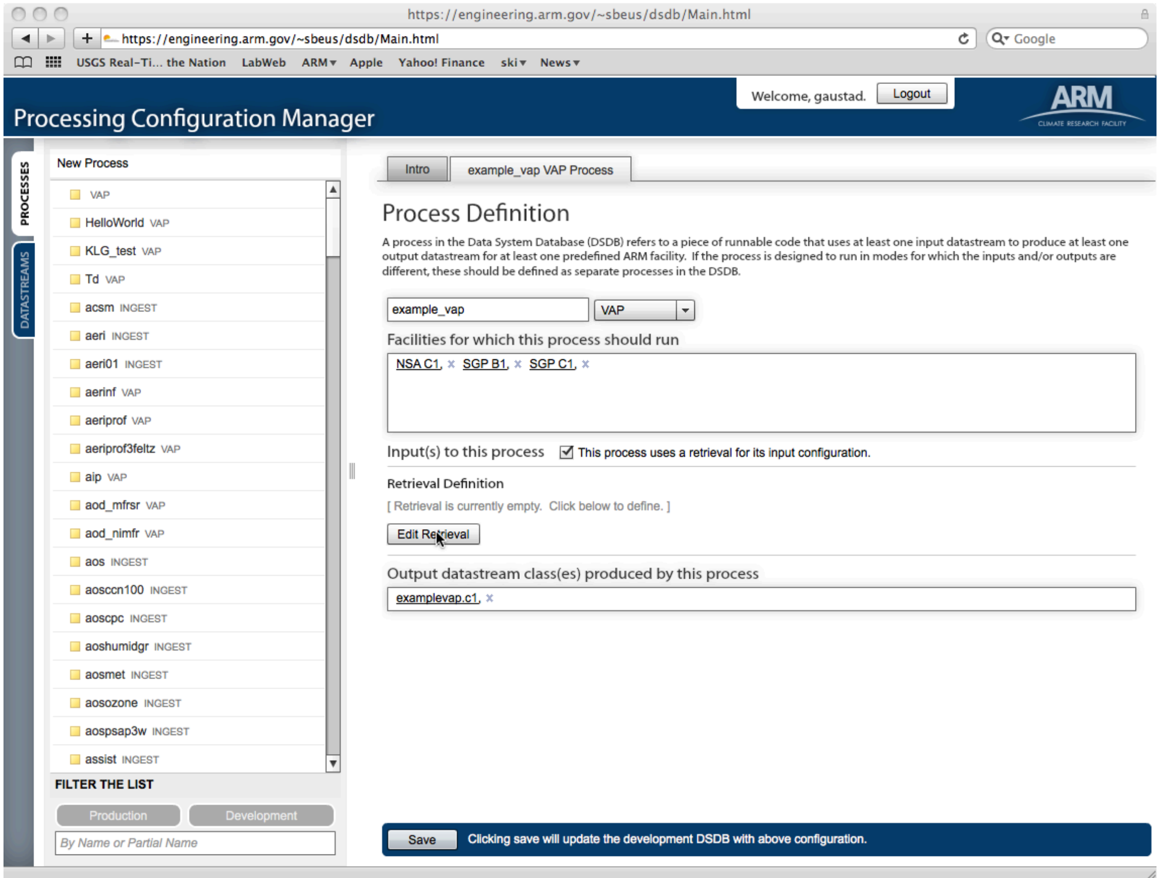


**Figure 2.5. PCM Process Definition Tool with Edit Retrieval Button Selected**

# 3 Define Variables to Retrieve (Retrieval Definition Tool)

At the most basic level, defining the inputs to a VAP consists of documenting the name of a variable and the datastream from which it should be retrieved. Historically, a significant effort was frequently expended performing pre-analysis data consolidation and transformations to prepare input data for scientific analysis. To minimize, if not eliminate, the need for VAP developers to perform such tasks, ISDE allows a user to

- Define preferred and alternative input datastream sources
- Assign a generic name to retrieved variables that will be referred in the automated source code
- Use a simple check box to retrieve companion QC variables
- Apply unit and data type changes to the data as part of the retrieval process.

Additional control is also provided to define input data source preference by site/facility pairing or time range dependencies.

The inputs of all VAPs must be specified using a retriever process. As such, by default the 'This process uses a retrieval for its input configuration' box will be checked and an 'Edit Retrieval' button should be evident in the lower left side of the right panel (see Figure 2.5). Selecting this button will bring up the Retrieval Definition form shown in Figure 3.1. Note that Retrieval Definition form has replaced the Process Definition form but it is still organized under the 'example_vap VAP Process' tab.

To return to the main Process Definition form select the 'Done' button at the bottom left of the right panel.

**Note**: Retriever data will not be stored in the DSD until the 'Save' button in the Process Definition form has been selected.

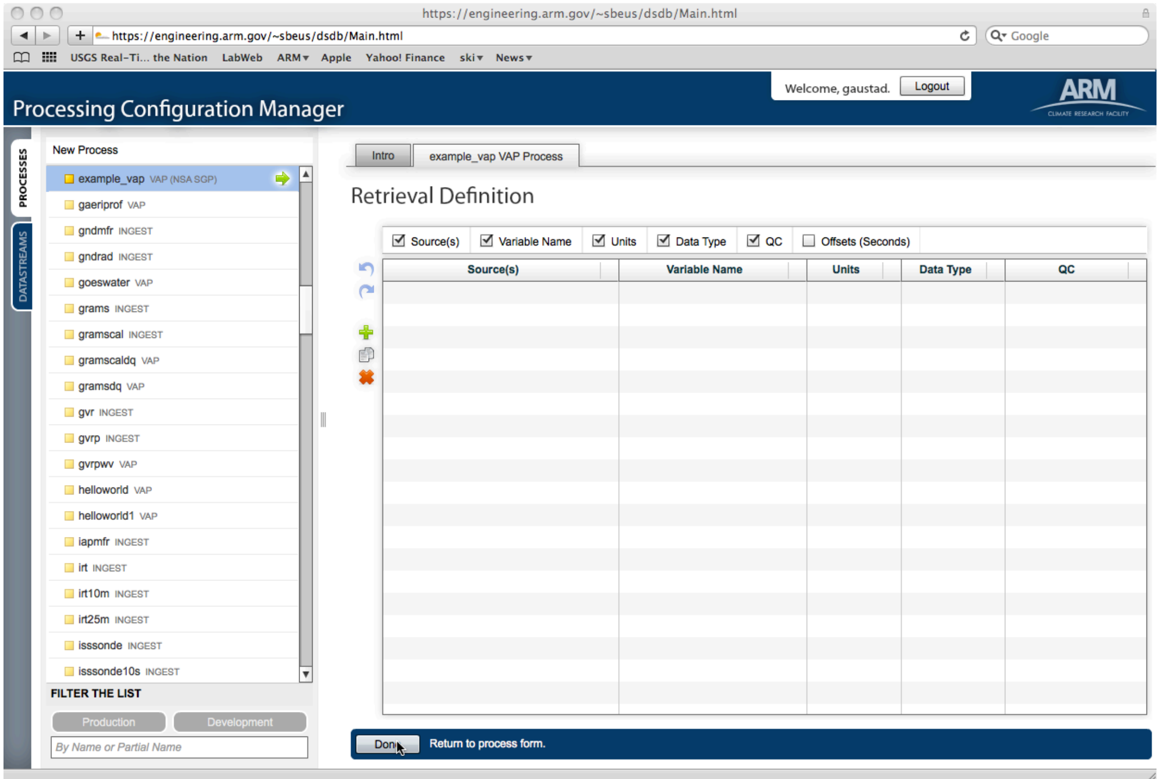


**Figure 3.1. Retrieval Definition Form**

## 3.1 Retrieval Definition Table Overview

The Retrieval Definition form allows the user to not only specify the variable and data source from which to retrieve a variable, but also to perform some basic transformations of units and data type. These options are checkboxes in the bar above the table. Selecting an item adds the data entry column to the table. In addition to the transformations, the bar also allows the user to retrieve data for a particular variable for some extra time before and/or after the process period specified in the command line, and to automatically retrieve the companion QC variable. A description of each of the columns in the Retrieval Definition table is given below.

**Variable Name:** Required.

- The Variable Name consists of the user defined name of the variable to be retrieved, and an indication of whether finding the variable in one of the specified input data sources is a requirement that must be met for the VAP to successfully run.
  - Variable names in the 'Variable Name' column must be unique.
  - If the 'Required' check box is marked, the VAP process will fail to run for a given observation (i.e., input data file) unless the variable specified is successfully retrieved
  - If the 'Required' check box is marked, an asterisk will follow the Variable Name.
  - This will be the name to which the retrieved data will be referred to in the DSDB and in auto-generated code.

- It is not necessary for this name to match the name in the datastream(s) from which the variable is retrieved.
- Coordinate dimension variables (i.e., time, height, range, etc.) should not be included in the Retrieval Definition table, as all coordinate dimensions of retrieved variables are automatically retrieved. This automatic retrieval is only successful when the dimension name and variable name in the input datastream file are identical.

**Units:** Optional

- Specifies the units into which the retrieved data will be converted. Units are converted using Unidata's UDunits library.
  - DEFAULT value results in units staying the same as found in the input file from which the variable is retrieved.
  - Units are entered free form. Please reference Unidata's web page for further information: http://www.unidata.ucar.edu/software/udunits/udunits-2/udunits2.html.

**Data Type:** Optional

- A drop list of possible data types into which the retrieved data can be converted.
  - If a value is provided the data type will default to type *float*.
  - If the data type remains as a default value through the population of the Data Sources Definition form, and a field is selected from the drop list of available values, the data type will be updated to the type of the selected field as found in the specified datastream.
  - If the default value is overridden in the Retrieval Definition table, the data type will not be updated as a result of field selections in the Data Sources Definition form.

**QC:** Required

- Indicates whether the companion QC variable will be retrieved in addition to the variable noted in the Variable Name and whether if successfully finding the companion QC variable is a requirement for the VAP to run.
  - It is assumed that the companion QC variable will be equal to the name of the variable in the input datastream file preceded by 'qc_'.
  - If the 'Required' check box is marked, the VAP process will fail to run for a given observation (i.e., input data file) unless both the variable and its QC is successfully retrieved.

**Sources**: Required.

- Datastream Source(s) is the datastream(s) from which the value(s) for the variable should be retrieved. It is populated via the Data Sources Definition form. A single value can be retrieved from a prioritized list of preferred and alternate datastreams. (in Figure 3.2 the first_cbh variable is retrieved from either the vceil25k.a1 or vceil25k.b1 based on the indicated conditions and correlated to a user defined variable 'cloud_base_height').

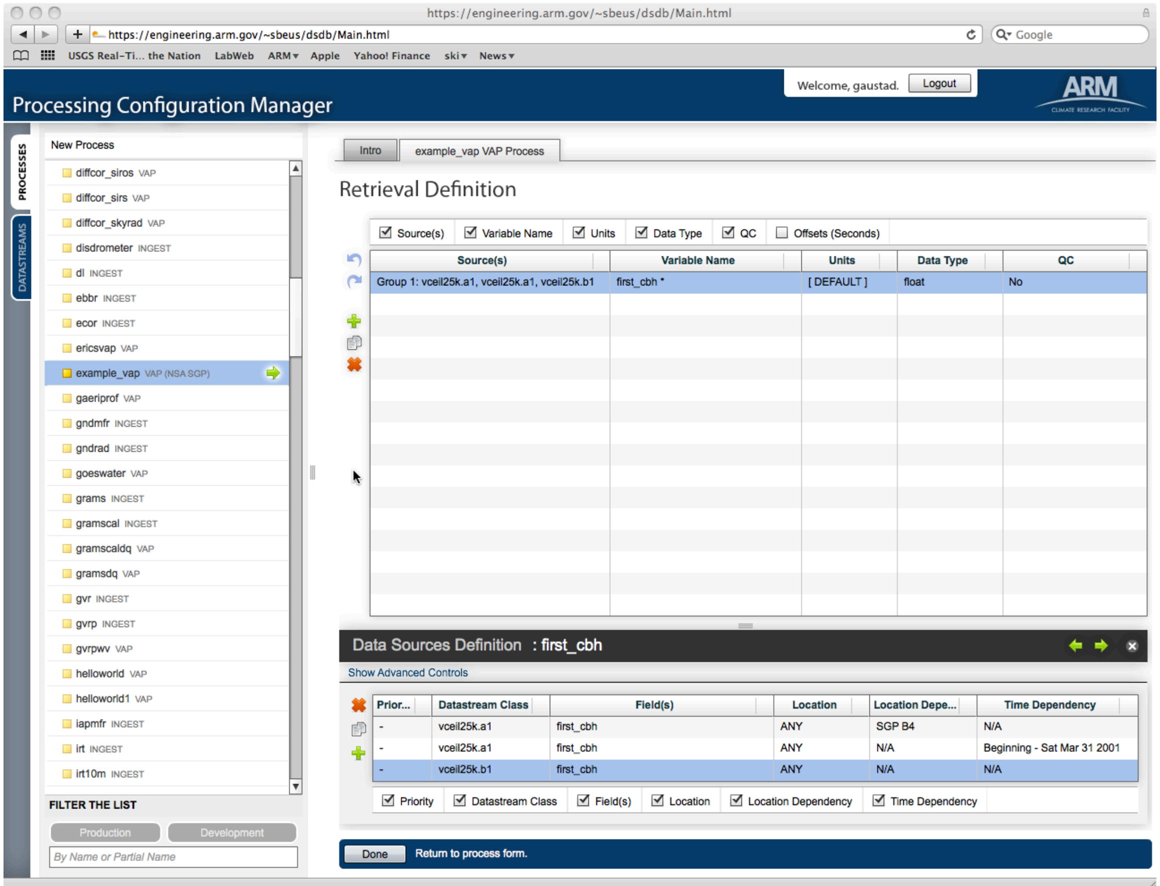


**Figure 3.2. Example of Retrieving a Single Value from the Option List of Datastreams with Location and Time Dependencies**

## 3.2 Data Sources Definition Form Overview

The Data Sources Definition form allows a user to define the source(s) of the data to retrieve and assign to the user defined variable. It allows for lists of preferred and alternate data sources, multiple possible variable names, and location and time dependencies. A description of each of the columns in the Data Sources Definition form is given below.

**Priority**: Optional.

- Integer representation of priority when alternative Datastream Sources are specified.
  - When priority is not populated, the first row is the highest priority and the last is the lowest.
  - Dragging and dropping the rows into the desired order is another way to adjust priority.

**Datastream Class:** Required.

- Datastream from which the variable with the name noted in the 'Field(s)' column will be retrieved.
    - Must be populated first before any of the other elements in the Data Sources Definition form can be populated.

**Field(s):** Required.

- Name of the variable to retrieve as found in the datastream defined as the Datastream Class.
    - Initially populated with a default value equal to the user defined Variable Name from the Retrieval Definition form. This default value is noted by brackets.
    - Value defaults to the user defined Variable Name in the Retrieval Form.
    - If the datastream is loaded into the history database, clicking on the Fields cell will bring up a drop list populated with all possible variable names. If not, the user should enter the desired variable name followed by a <return>.
    - If more than one variable is entered into the Fields column, the retriever searches the input datastream file for each of the variables in the order listed, until one is found.

**Note**: The variable names shown in the drop list reflect all the variables that have existed for that datastream over all time, not just the variables in the datastream's latest DOD.

**Location:** Optional.

- For use when the datastream from which to retrieve data is independent of the location at which the VAP process is running. For example, if a VAP will always retrieve the temperature from the sgptwrmrC1.c1 datastream, even when running at other site/facility pairings, select twrmr.c1 as the Datastream Class and 'SGP C1' from the Location drop list.

**Location Dependency:** Optional.

- Used when the datastream from which to retrieve data is a function of the site/facility at which the VAP is being run.

**Time Dependency:** Optional.

- Used when the datastream from which to retrieve data is a function of what period the VAP process is running.
    - If a *begin* or *end* time dependency is not selected, the time dependency defaults to the beginning of the datastream or end of the datastream respectively.

An example of both a location and time dependency is illustrated in Figure 3.2. In this example, when the VAP is run for sgpB4, the user defined variable 'cloud_base_height' will be correlated to the first_cbh variable in the vceil25k.a1 datastream. If it is not running at sgpB4 and the date being processed falls before April 1, 2001, the user defined variable 'cloud_base_height' will be correlated to the variable 'first_cbh' in the vceil25k.a1 datastream. For process times April 1, 2001 or greater, and when processing at sites other than sgpB4, the user defined variable 'cloud_base_height' will be correlated to the first_cbh variable in the vceil25k.b1 datastream.

## 3.3 Populating the Retrieval Definition Table

Retrieval Definition table variables and data sources can be populated by either:

- Specifying the variable names and datastream sources by typing in the fields in the Retrieval Definition Table.
- Accessing the DOD of an existing datastream using the Datastreams tab on the left panel of the Process Configuration Manager and dragging and dropping variables to retrieve into the table.

Manual entry is more efficient where there is more than one datastream from which to retrieve the variable. When a variable's source is a single datastream (no alterate sources if that datastream is unavailable), it is more efficient to access the DOD of the input datastream and drag and drop variables onto the Retrieval Definition table.

1. To enter the Retrieval Definition form.
   a. From the Process Definition Tool (Figure 2.4) select the 'This process uses a retrieval for its input configuration' button.
   b. Select the 'Edit Retrieval' button.
2. Populate the Retrieval Definition Table.

   **Note**: Do not retrieve netCDF standard (lat, lon, alt) or coordinate dimension variables (time, height, range) for a retrieved variable as these will be automatically retrieved.

### 3.3.1 Manual Entry of Input Data Variables and Sources

a. Select the green plus symbol located to the left of the table form.

b. Select the 'custom_field_1' variable and enter the name of the variable to retrieve.

c. Indicate whether the variable must be found for the VAP to run via the 'Required' check box.

d. Select 'Source(s)' [NONE] in the Datastream column

e. Select the pencil icon to bring up the Data Sources Definition form.

f. Select the Datastream column corresponding to the Field with the value of the variable you just defined to bring up a drop list of possible datastreams.

   If the data source is a single datastream with no alternative sources:

   - Select the datastream from which the variable should be retrieved then proceed to step 'g'.

   If the data source is a single datastream with alternative sources based on datastream availability:

   - Select the most preferred datastream from which the variable should be retrieved then proceed to step ii.

   If the data source is a single datastream with alternative sources based on location or time dependencies:

   1) Select the datastream from which the variable should be retrieved and define the most restrictive dependencies.

2) Create a new row in the Data Sources Definition table by either selecting the green plus symbol to the left of the table, or by duplicating an existing row in the table by selecting the paper symbol on the left.
3) Update the Datastream Class column of the new row to reflect the next most preferred (or next most restrictive) data source.
4) Repeat the addition of new rows until options are exhausted.
5) Review data source priority and populate the 'Priority' column as necessary.

g. If the name of the variable found in the Datastream Class datastream does not match the default value, update the entry in the 'Field' and select the desired variable name.

h. If a second value is to be retrieved and correlated to the user defined Variable Name in the Retrieval Definition form (meaning the user defined variable in the Retrieval Definition form will be an array of more than one value) specify the data sources and associated values of the additional values as follows:

1) Select the 'Show Advanced Controls' button in the upper left corner of the 'Data Sources Definition' window. This will bring up additional icons along the top left of the Query window to add, close, delete, and adjust the order of additional queries.
2) Define a new query to retrieve the additional data value by selecting either the green plus or the paper sheet icon to add a new or duplicate query.
3) For the new query define the data source(s) and update the field name.

i. Close the Data Sources Definition window by selecting the 'x' in the upper right corner of the window.

j. Return to the Retrieval Definition form and specify additional variables to retrieve by adding new rows to the Retrieval Definition table.

### 3.3.2 Dragging and Dropping Input Data Variables and Sources from Existing DODs

You can populate the Retrieval Definition Table by Dragging and Dropping from Input Datastream DODs. For example, from the datastream and the variables you can drag and drop the desired variable into the Retriever Definition table. The Source(s), Variable Name, and QC retrieval status will be populated. Update these as required.

a. Select the Datastream tab and locate the datastream from which to retrieve the variable.

b. Select the triangle next to the highest DOD version of the desired input datastream (for example the DOD) to list the dimensions, variables, and global attributes associated with the DOD.

c. Select the triangle next to the Variables to expand the variables.

d. Select the variable to retrieve it with a single click and drag the variable from the left frame and drop it into the Retrieval Definition frame on the right (Figure 3.3).

e. Update the variable's Source(s), Variable Name, Units, Data Type, QC, and Offset values as necessary.

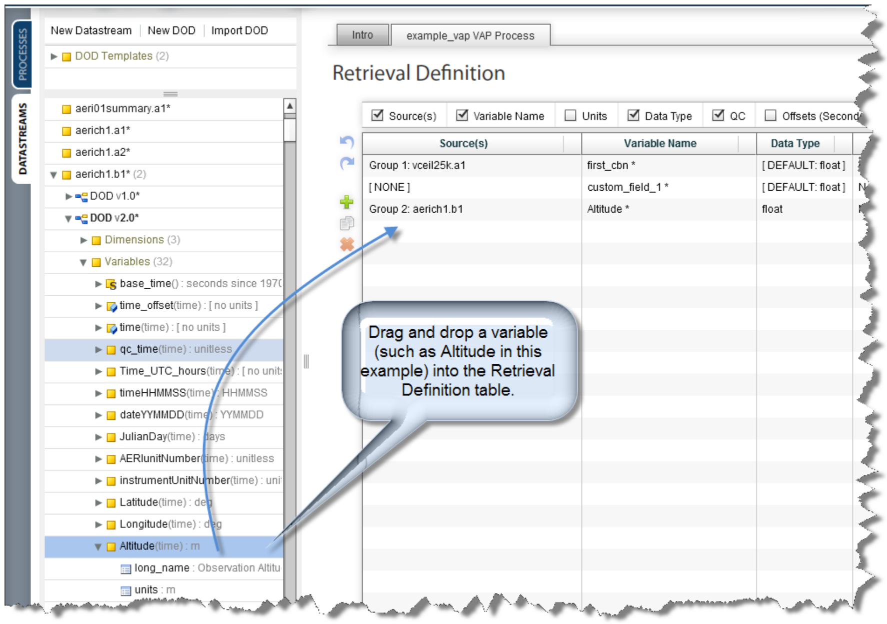


**Figure 3.3. Dragging and Dropping a Variable**

For the example VAP that will be developed in this tutorial, we will retrieve first_cbh, qc_first_cbh, and backscatter variables from the vceil25k.b1 datastream. The first_cbh will be saved into a user defined variable name of 'cloud_base_height' and written to the output netCDF file with that name. If that datastream is unavailable the variables will be retrieved from the vceil25k.a1. The units of the first_cbh will be converted to centimeters and the successful retrieval of the QC variable will be required for the example_vap process to run. The Retrieval Definition table for example_vap, with the Data Sources Definition form open for the fist_cbh variable, is shown in Figure 3.4.

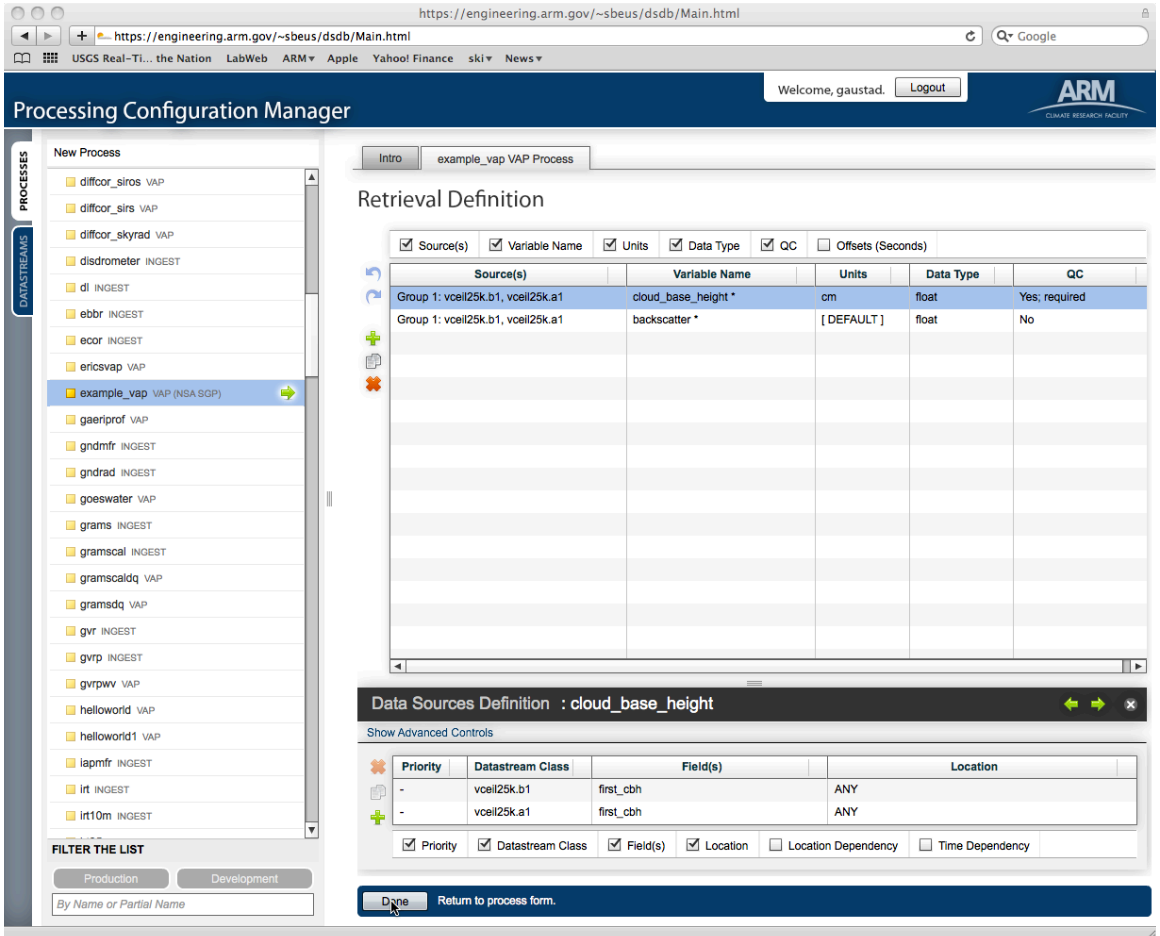


**Figure 3.4. Data Sources Definition Form for example_vap VAP**

The duplicate entry icon (paper sheets icon) in the Retrieval Definition form was used to add the backscatter variable since it is retrieved using the same Data Sources Definition query (i.e., sets of possible input datastreams). The duplicate entry was updated as appropriate for the backscatter variable (update the Variable Name, Field(s), and Units). The coordinate dimensions of the retrieved variables (time and range) and lat, lon, and alt are not included in the Retrieval Definition table as they will be automatically retrieved. Note that the Location Dependency and Time Dependency check boxes in the Data Sources Definition have been deselected as they are not applicable to this example.

## 3.4 Saving the Retrieval Definition to DSDB

The input data retrieval specifications are saved to the DSDB from which it is accessed by the isdetemplater application to create the C project source code files used at run time by the VAP. Note in Figure 3.5 that the user defined variable names that are retrieved from the input datastreams are summarized above the 'Edit Retrieval' button.

To save retrieval data to the DSDB, perform the following steps.

a. Select the 'Done' button in the lower left corner of 'Retrieval Definition' form to return to the 'Process Definition' form (Figure 3.4).

b. Select the ‘Save’ button in lower left corner of ‘Process Definition’ form.

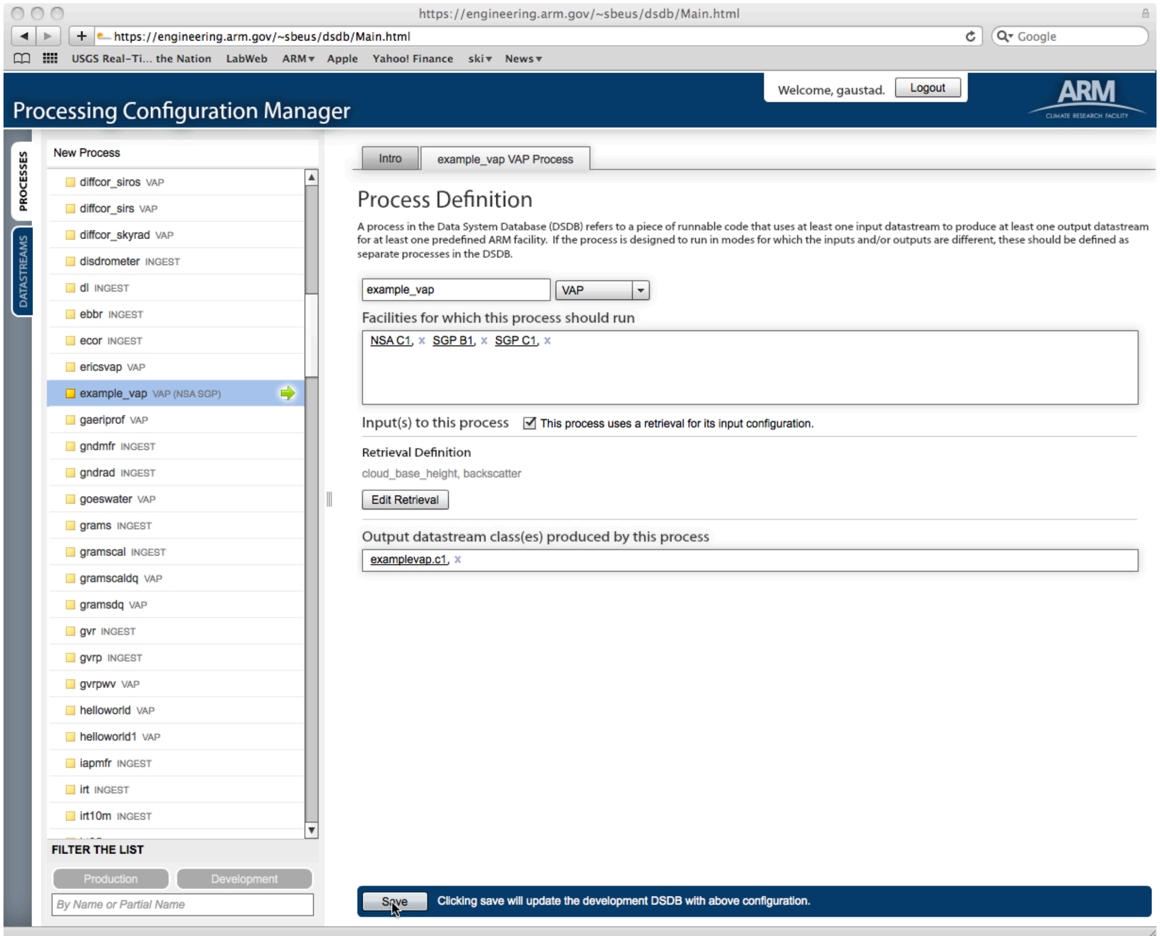


**Figure 3.5. Completed Process Definition Form for example_vap VAP**

# 4 Birth of a Datastream Documentation (Datastream Information Tool)

Birth of a Datastream (BOD), or the Datastream Information Tool, is a collection of specifications used by the archive to describe ARM data products. This form should only be populated for VAPs that are to be released to the ARM production processing system. A developer responsible should work with their translator on populating this form. Figure 4.1 shows the Datastream Information page for the examplevap.c1 output datastream of the example_vap process.

**Note**: At the top of the Datastream Information page, the example_vap process tab is still open and available.

**CAUTION**: Do not perform this step unless you are working with an ARM science translator on a production ARM VAP.

## 4.1 Populating BODs

To populate the BODs page for an output datastream(s), perform the following steps.

a. Select the Datastreams tab in the upper left corner of the left frame of the Process Definition tool to replace the list of processes with a list of datastreams (Figure 4.1).

b. Scroll down and find the newly created datastream(s) and double click on it to load the Datastream Information form into the right panel of the web page.

c. Work with the VAP translator to populate the form.

d. Select the 'Save' button at the bottom of the page to save entries to the datastream database (DSDB).

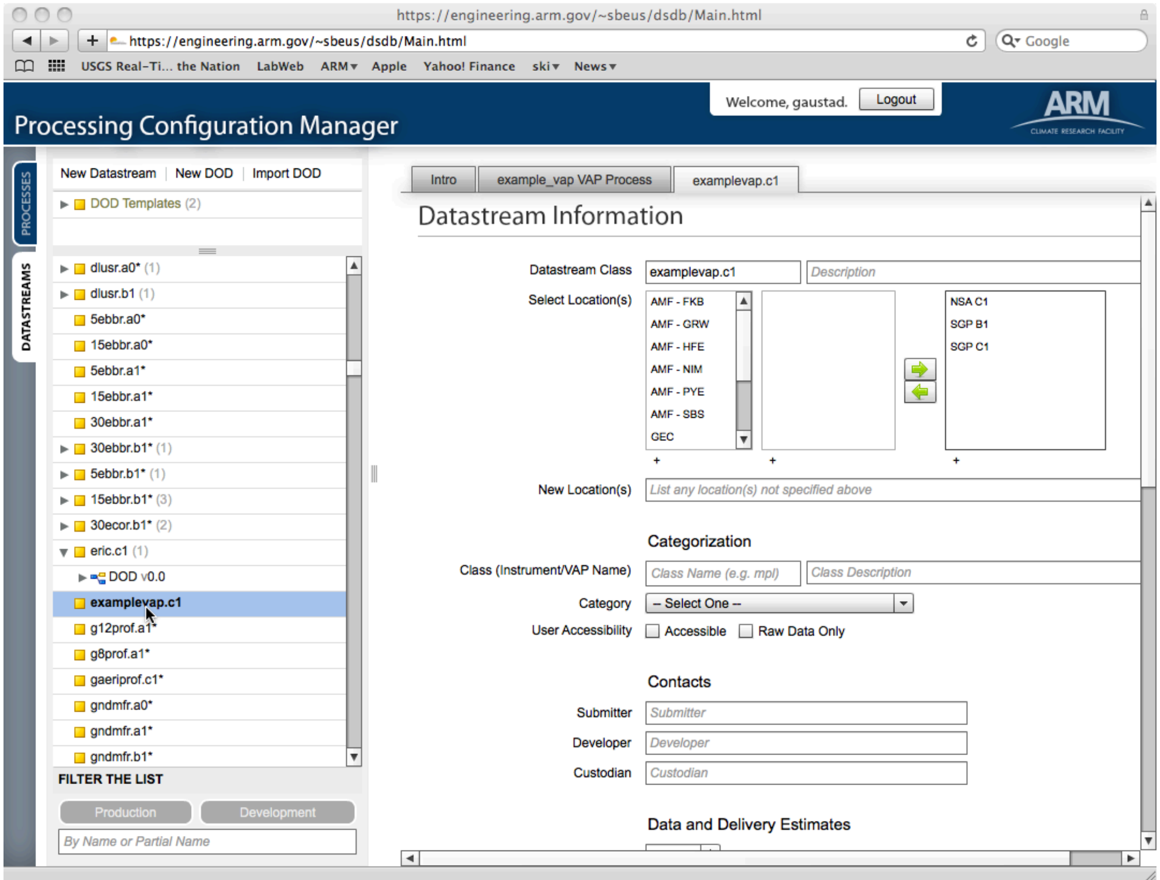


**Figure 4.1. Datastream Information, BODs Document, for the examplevap.c1 Datastream**

# 5 Define Content of Output Datastream(s) (DOD Configuration Tool)

For an ISDE VAP to run the DOD of all of its output datastreams need to be loaded into the DSDB. A new DOD can be created by either starting from a blank template or by importing an existing netCDF file. If importing a DOD from a file, the file must follow arm data file naming standards, and the site and facility of the datastream must be loaded into the DSDB. Once loaded into the DSDB, the DOD(s) should be updated to conform to ARM DOD standards (add reference once it exists).

DODs associated with a datastream can be accessed via the 'Datastream' tab located in the left frame of the PCM. Datastream's with DODs loaded into the are noted with a triangle that if selected, expands to list the datastreams. As shown in Figure 4.1, the examplevap.c1 datastream does not currently have a DOD associated with it.

The process of completely defining a datastream's DOD includes creating the DOD, adding variables, adding/editing variable attributes, adding global attributes, and resetting variable attribute and global attribute values that will be populated by the ISDE libraries or the VAP algorithm. All of these steps will be required for VAPs with DODs created through the use of a template. Whether, and if so, how much a DOD loaded by importing from an existing netCDF file will need to be edited is a function of the contents of the netCDF file imported and whether the DOD meets ARM's standards.

## 5.1 Creating a DOD

To create a DOD you can either create a new one from a blank template, or by importing one from a netCDF file. We discuss both techniques in the following steps.

### 5.1.1 Creating a New DOD from a Blank Template

a. Select the 'New DOD' cell located in the center column at the top of the left frame.

b. Select the 'template.c1' from the DOD Template window (see Figure 5.1) to create a DOD with the dimensions, variables, and global attributes commen to all VAP processes (Figure 5.2).

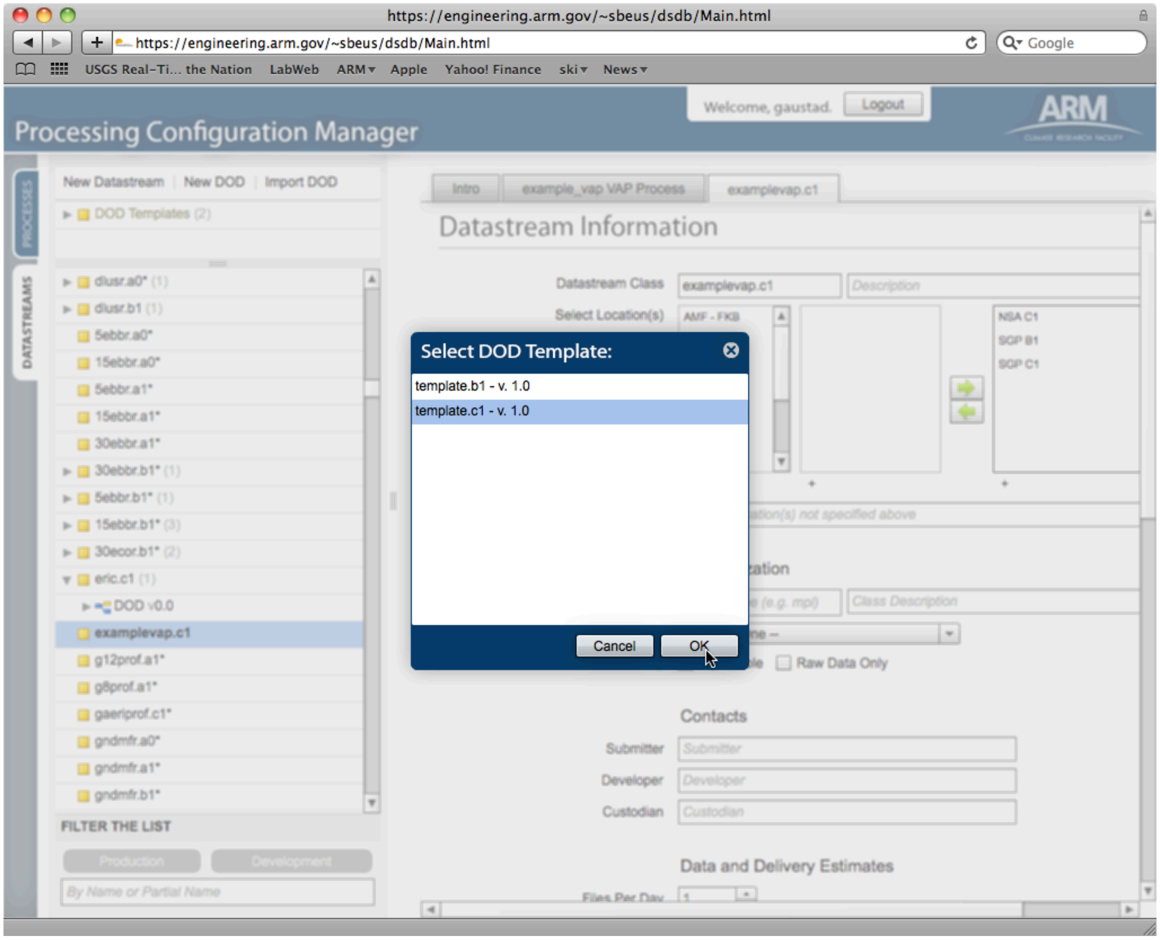


**Figure 5.1. 'Select DOD Template' Window**

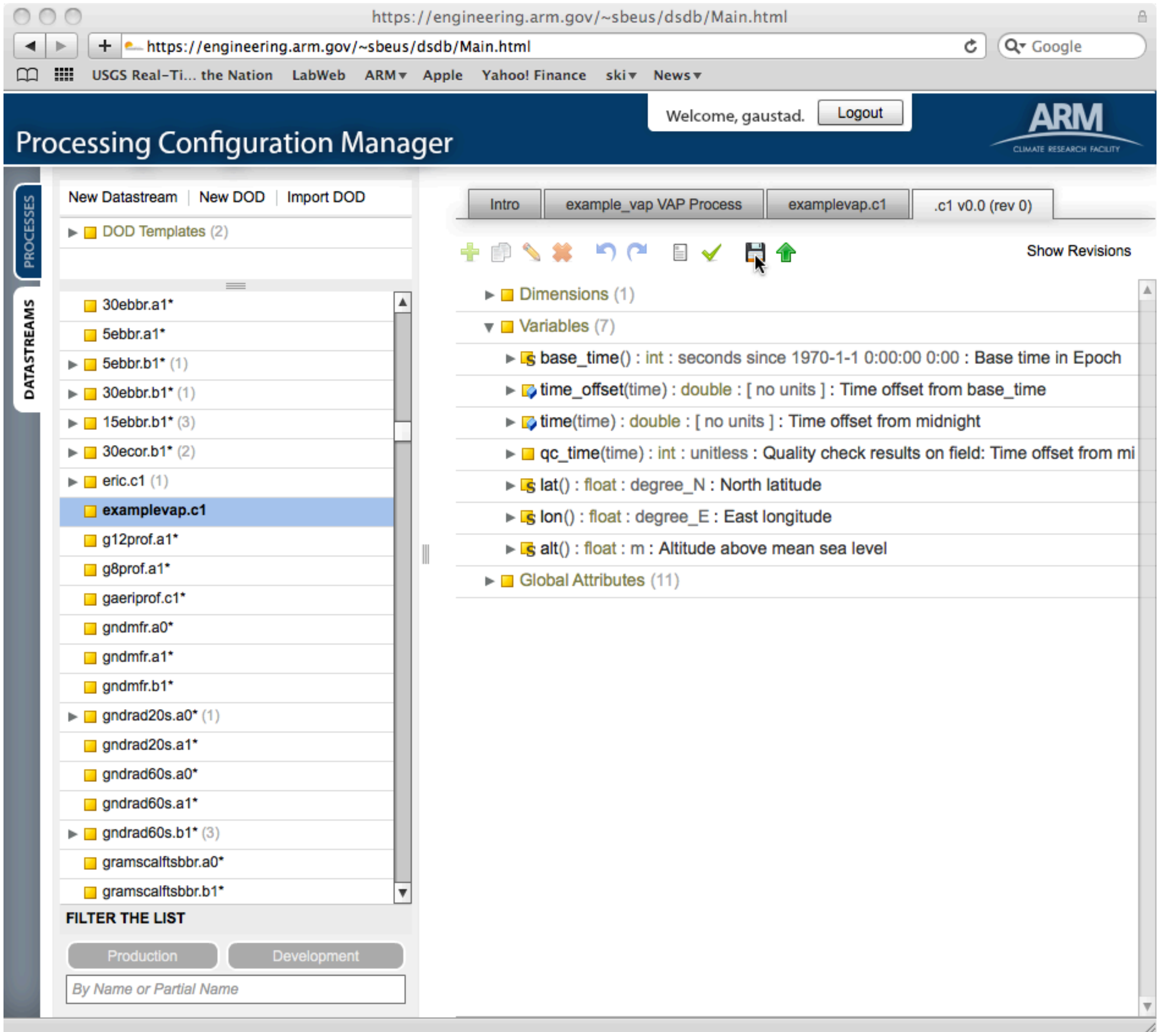


**Figure 5.2. DOD VAP Template**

## 5.1.2 Creating a New DOD by Importing from a NetCDF File

a. Select the 'Import DOD' cell located in the right column at the top of the left frame.

b. Select the 'Browse Local' button to find the netCDF file located on your local computer to use a template for the VAP's DOD (see Figure 5.3).

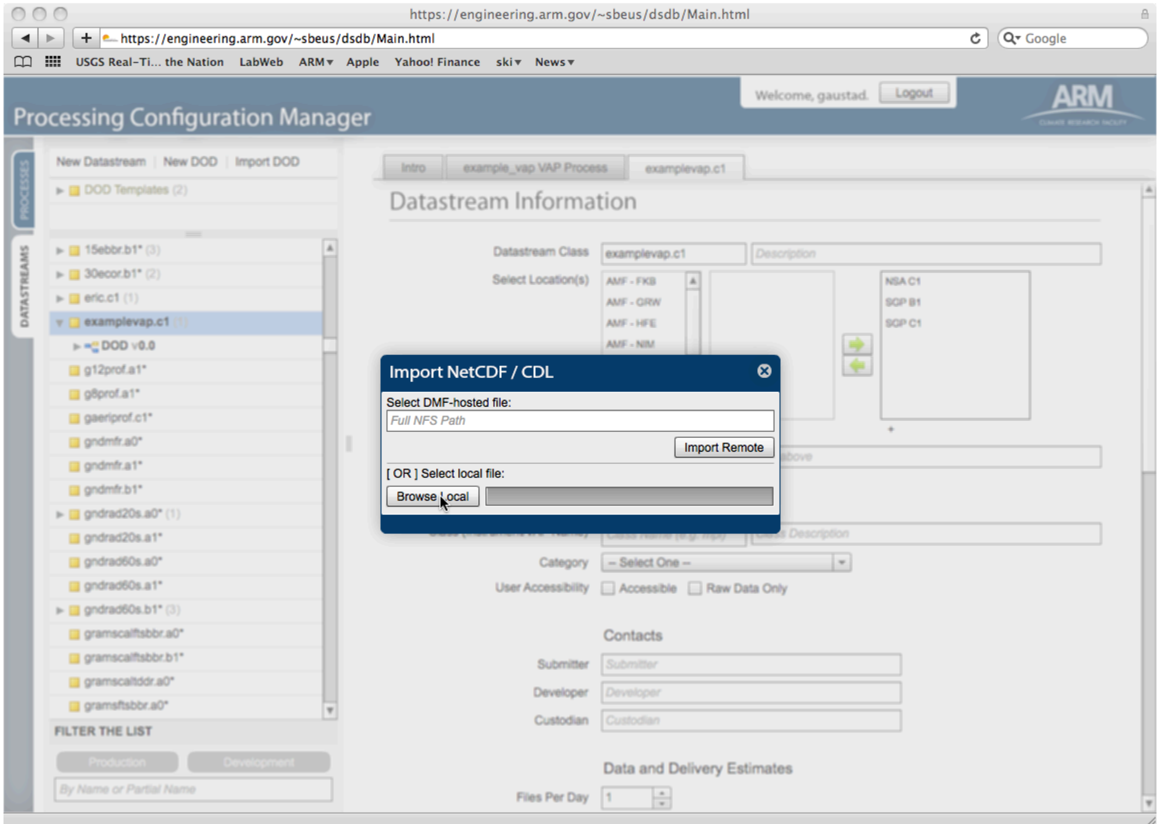


**Figure 5.3. Import DOD from netCDF file on Local Computer**

1. Saving a DOD

   a. Select the disk icon located at the top of the DOD (Figure 5.2) to save the DOD to the DSDB.

   b. Enter the output datastream name (base platform name and data level) of the datastream to which the DOD will belong.

   c. Enter the version of the DOD, and a comment describing the DOD (Figure 5.4).

   **Note**: more than one DOD version can be defined for any given datastream.

   d. Select the Save button to save the DOD to the DSDB.

   **Note**: If an inappropriate name is selected the datastream name and level box shown in Figure 5.4 will be highlighted in red and the DOD will not be saved.

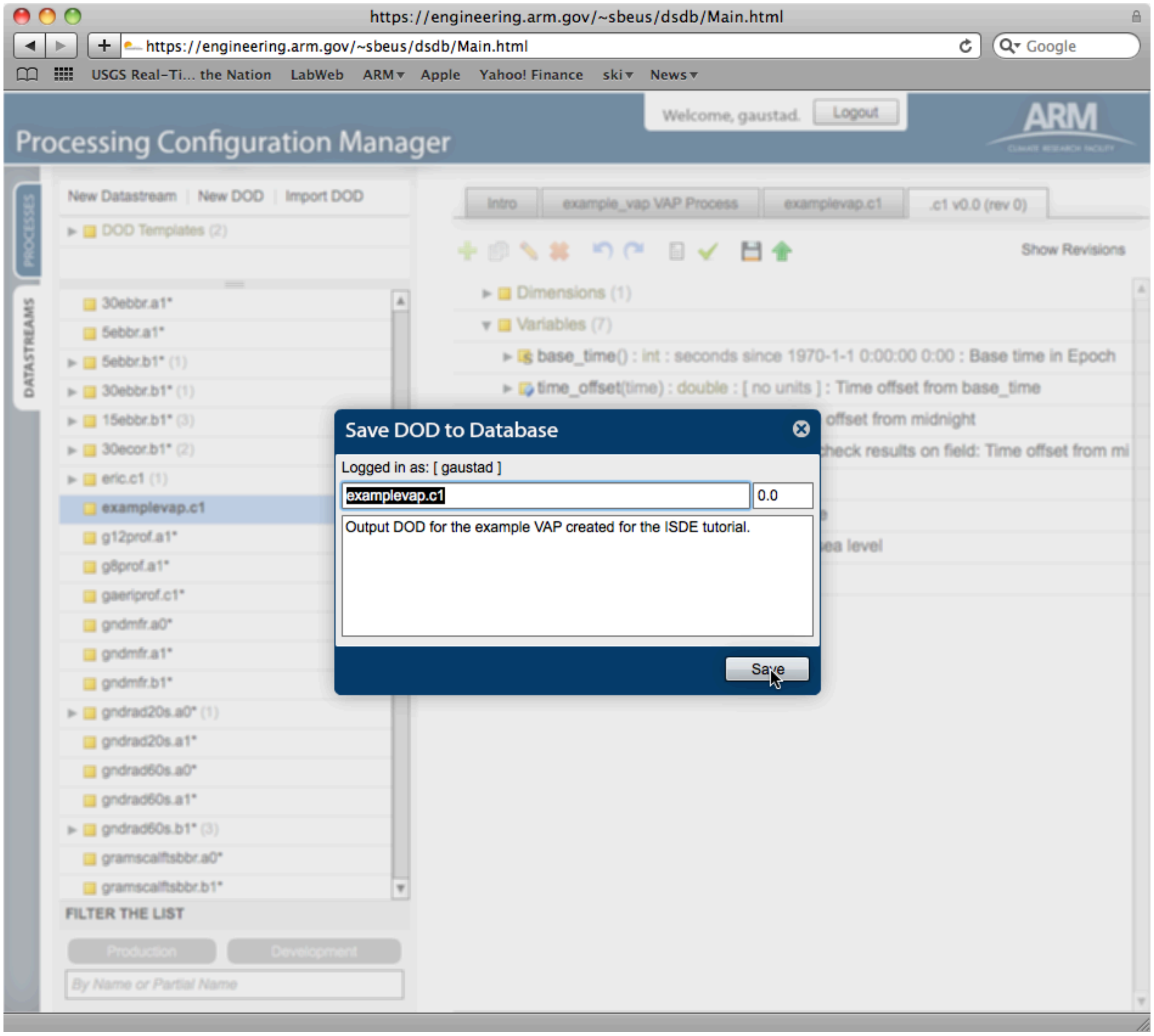


**Figure 5.4. Save DOD to DSDB**

Two versions of a DOD, for examplevap.c1 datastream, were loaded and saved in the previous two steps. Version 0.0 was created via the 'New DOD' button using a default template, and V0.99, which was created by importing the header of a vceil25k.b1 data file. The steps of editing to create the desired output data product for the example_vap process were roughly the same, but the imported DOD will need to have most of its variables deleted in addition to update the attributes of those that remain.

## 5.2 DOD Elements

**Dimensions**:

- All dimensions that comprise a variable's coordinate system must be defined in the DOD. A full definition includes not only meeting the netCDF requirement of documenting dimension name and length, but also, when possible, includes creating a variable whose name matches that of the dimension, whose dimension is the dimension itself, and whose values document the possible values of that element of another variable's coordinate system. For example, if a measurement in a file is dimensioned by time and height, where time is the unlimited dimension and height has a value of 10, a variable named 'height' should be created that records the 10 heights of the temperature variable's coordinate system.

**Variables**:

- Output variables include not only variables representing the primary measurements produced by the VAP analysis, but also all input variables whose values are used in determining the primary measurements. These variables are referred to as 'passthrough variables'.
- Variable attributes that must be defined for all variables include the variable name, dimensions, data type, long_name, and units. Attributes that should exist if they are applicable to the variable include valid_min, valid_max, valid_delta, resolution, and missing_value. Optional attributes that can be added to provide additional detail include: comment(s), precision, accuracy, and uncertainty. In addition to this attributes, an indication of whether the variable has a companion QC variable and whether it is a primary measurement can also be made.

**Global Attributes**:

- Global attributes are metadata that apply to all variables in the file and over all time samples. Typically metadata should not have attributes attributed to it because it is an attribute. As a result, if a global attribute has a unit then it should probably be defined as a static variable rather than a global attribute so that its units can be properly defined in a variable attribute.

## 5.3 Adding DOD Elements to a DOD

Variables, dimensions, variable attributes, and global attributes can be added to a datastream's DOD by either adding a new DOD element or by dragging and dropping DOD elements from an existing datastream's DOD. Frequently it is more efficient to add passthrough variables, and the variables correlating to their coordinate dimensions to a DOD by dragging and dropping the variable from its input datastream. The drag and drop method is helpful for any DOD element if it closely resembles an element in an existing datastream.

The following tools are useful for editing the DOD variable attributes:

- Green plus – insert new DOD element after selected element.
  **Note**: This can be used to insert a dimension, variable, or global attributes.
- Duplicate sheets – copies element. Copies highlighted existing element and insert copy of that element immediately after the existing element.
- Pencil – edits selected element.
- Red 'X' – deletes selected element.
- Arrow rotating counter clockwise – undo last edit
- Arrow rotating clockwise – redo last edit
- Single sheet – brings up DOD in text editor
- Green check symbol – attempts to automatically fix simple problems in a DOD.
- Disk – save DOD to DSDB

- Green arrow pointing up – exports DOD to text file.
- Drag and drop functionality – DOD elements can be dragged and dropped across and within DODs. Multiple elements can be selected using the shift and control keys and moved all at once.
- If a DOD element is selected (dimension, variable, attribute), a mouse right click will bring up a menu of editing commands which if selected will be performed against the selected DOD element.

### 5.3.1 Adding a New DOD Element to a DOD

**By inserting a New DOD Element.**

a. Select the element (such that it is highlighted) after which the new element will be inserted.

b. Select the green plus icon located just above the dimensions to create the new element and open up the element's edit form in a third frame on the left of the PCM (Figure 5.4 shows the DOD Variable form).

**By Dragging and Dropping from an Existing Datastream.**

a. Locate a similar element in the PCM's Datastreams tab in the left frame. For a passthrough variable, the value to copy is the retrieved variable in most preferred input datastream.

   **Note**: the Retrieval Definition form can be accessed from the VAP Process window tab located at the top of the PCM page. Return to the DOD window by selecting the DOD tab.

b. Expand the datastream, the desired DOD version, and variables of the input datastream (see Figure 5.5).

c. Select the element(s) to retrieve by highlighting them (either with a single click for one variable, shift-mouse click for a range of variables, control-mouse click for more than one non-adjacent variable) and drag the element from the left frame and drop it into the desired location in the Retrieval Definition frame on the right.

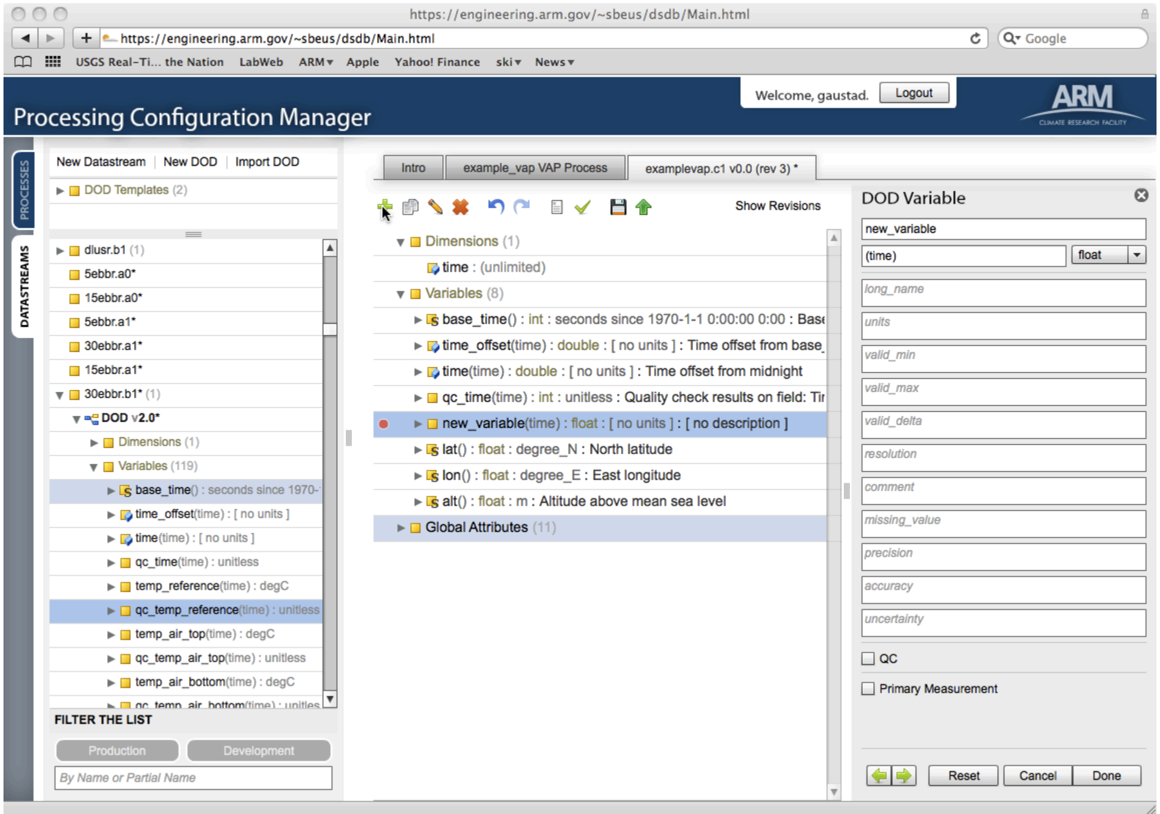


**Figure 5.5. Adding New variable Using New DOD Element and the Green Plus Icon**

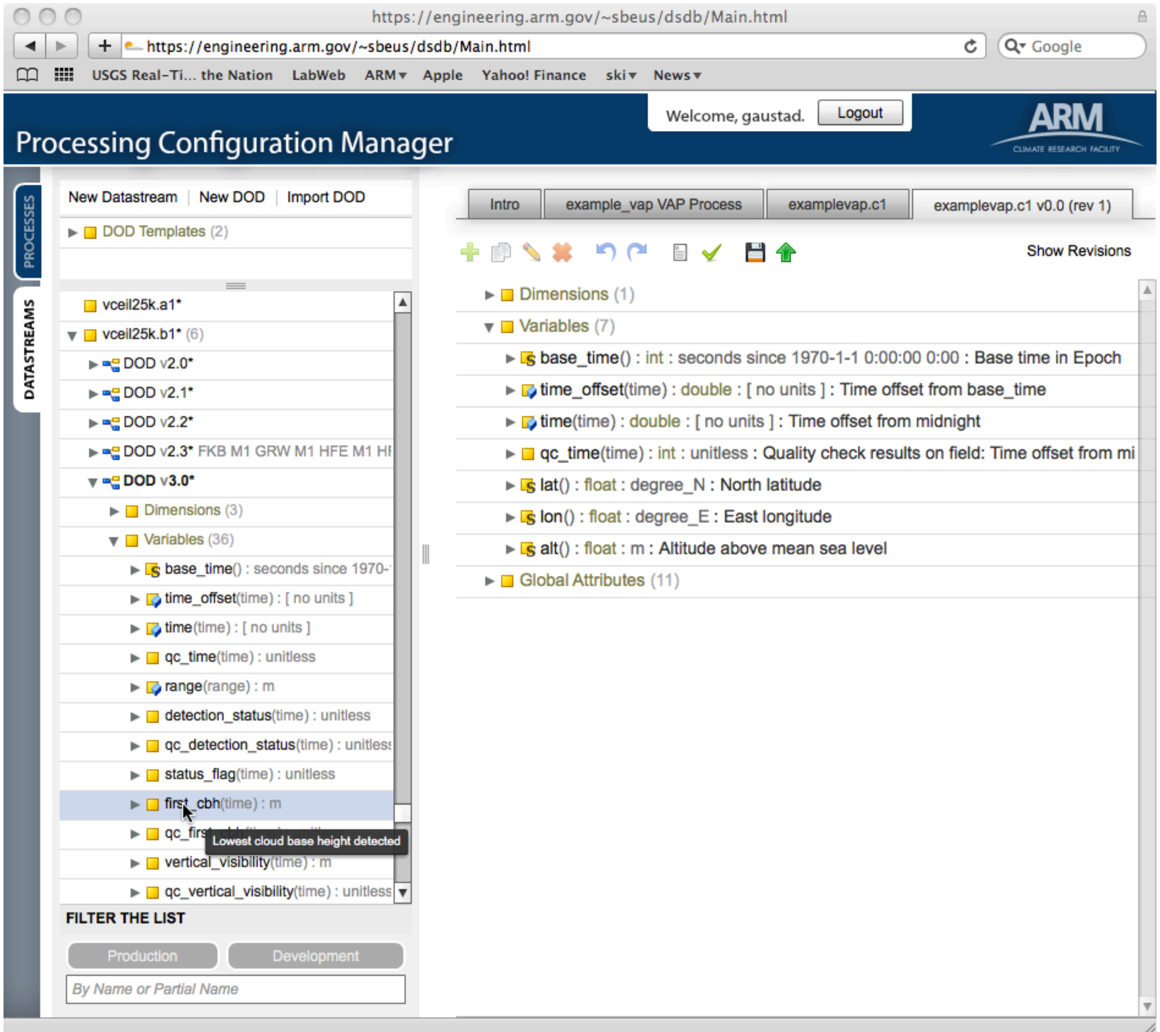


**Figure 5.6. Locate Passthrough Variable in Retrieved Variables Most Preferred Datastream**

### 5.3.2 Review Your DOD Elements

All elements and their attributes, regardless of how created, (imported, as a new element, or copied from an existing datastream) should be reviewed to ensure they meet ARM standards and have been properly defined. The primary measurement variable check box should be selected for all primary measurements so that this characteristic will be stored in the DSDB. Section 2 discusses variable adding and editing variable attributes in context of the drag and drop example, but the updates for a variable created from scratch are done in a similar manner.

### 5.3.3 Add/Edit DOD Elements

a. Access the element's edit form (by either double clicking on the element's name in the new DOD, or highlight the element and select the pencil icon located at the top of the page), or the elements DOD Attribute form (select the attribute and either double click or pencil icon), or select variable or attribute, right click, and select edit.

b. Create and set the values of the element.

   **Note**: When editing the DOD Variable elements:

   - If you specify additional dimensions for a variable in the DOD Variable form, those dimensions will be automatically added to the DOD.
   - If the QC box is checked, a QC variable will be created when the DOD Variable form is exited (i.e., the 'Done' button is selected).
   - Select the primary measurement box if the variable is a primary attribute.

c. Select the disk icon located at the top of the DOD (Figure 5.2) to save the updated DOD to the DSDB.

For the example_vap process being developed, the passthrough variables were populated in the examplevap.c1 DOD v0.0 through a drag and drop process. Figure 5.7 shows these variables expanded with their original values. Figure 5.8 shows the same variables after they have been edited for the change in name and units of the cloud base height variable, and to meet VAP QC standards. To expedite the process, the attributes of the qc_first_cbh were copied from another *c1 level datastream. Note that in Figure 5.7, the variables added to the basic template are noted in green, and in Figure 5.8, existing attributes that have been updated are indicated in red and new attributes are highlighted with a green bar.

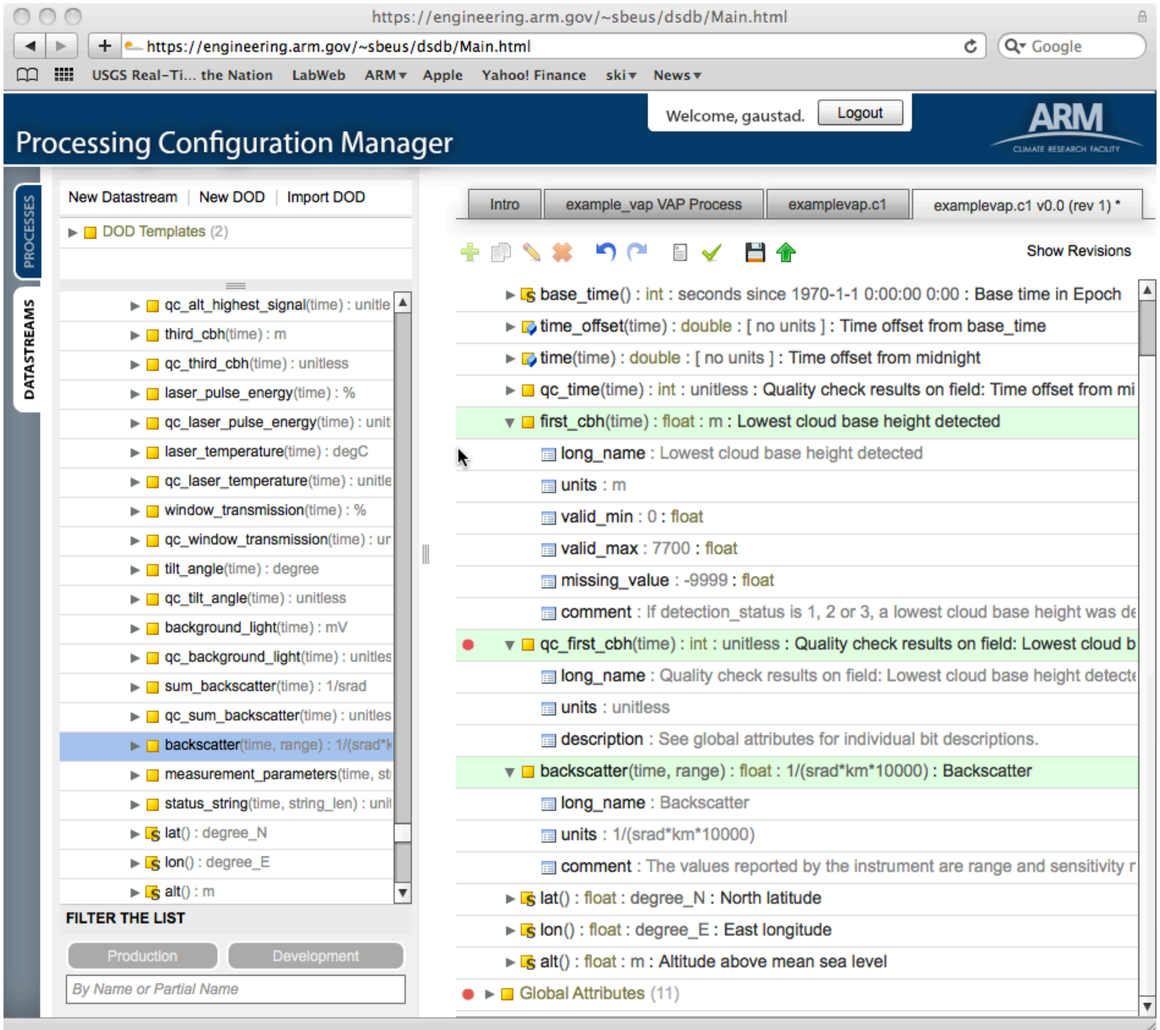


**Figure 5.7. Passthrough Variable Attributes Before Updates**

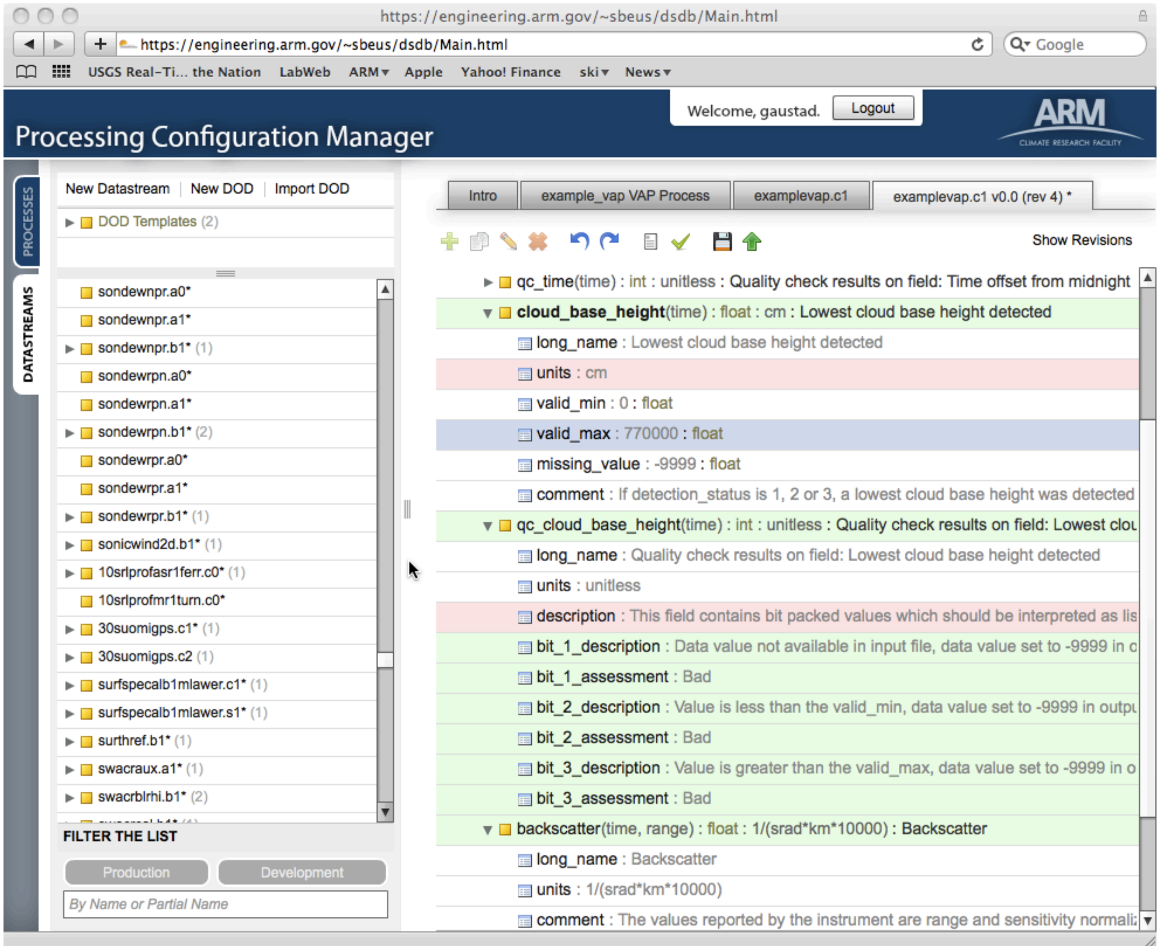


**Figure 5.8. Passthrough Variables After Editing**

For the example VAP we will add one new variable 'new_cloud_base_height' that is equal to the 'cloud_base_height' variable divided by 10 to convert its units to meters. The 'new_cloud_base_height' and 'qc_new_cloud_base_height' variables were created by duplicating the 'cloud_base_height' variable (select the cloud_base_height variable and then the paper duplicate icon). It is then edited to update its name and attribute values. The new values and their attributes are shown in Figure 5.9.

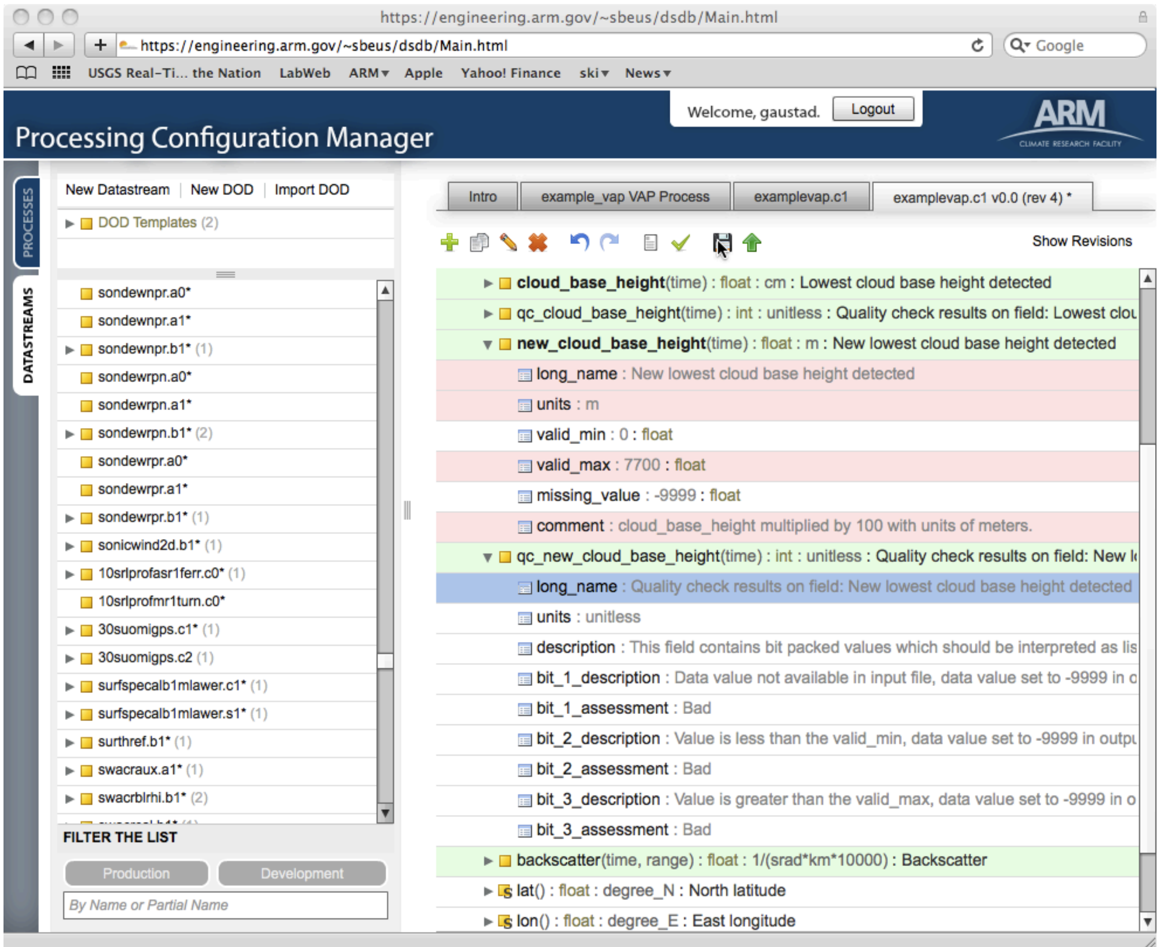


**Figure 5.9. Completed DOD for example_vap.process**

## 5.4 Fixing Problems in the DOD

Complete the definition of the Datastream by addressing the error, warning, and notice indicators in the DOD that appear to the left of the variables as red, yellow, and grey circles respectively. The colors indicate the confidence that a fix is needed and severity of leaving something unfixed. With red indicating it is highly likely a fix is needed and not fixing it could significantly impact a user's or automated processing application's ability to understand and utilize the information. Yellow indicates a fix is likely needed, and grey indicates a recommended change and not an indication of a problem. Mousing over the indicator will reveal a pop up describing the issue. If the pop up is '+ 1 child noice', expand the variable and the indicator will move to the appropriate attribute of the variable and the pop up will be updated with the specific issue.

In Figure 5.10 the error 'Missing or invalid coordinate dimension for this dimension' occurs because the 'backscatter' variable is dimensioned by time and range. While the range dimension was automatically added to the DOD, ARM standards require that also be a variable 'range' included in the DOD if it can be defined. This issue can be resolved by returning to the input datastream, and dragging and dropping the 'range' variable into the DOD.

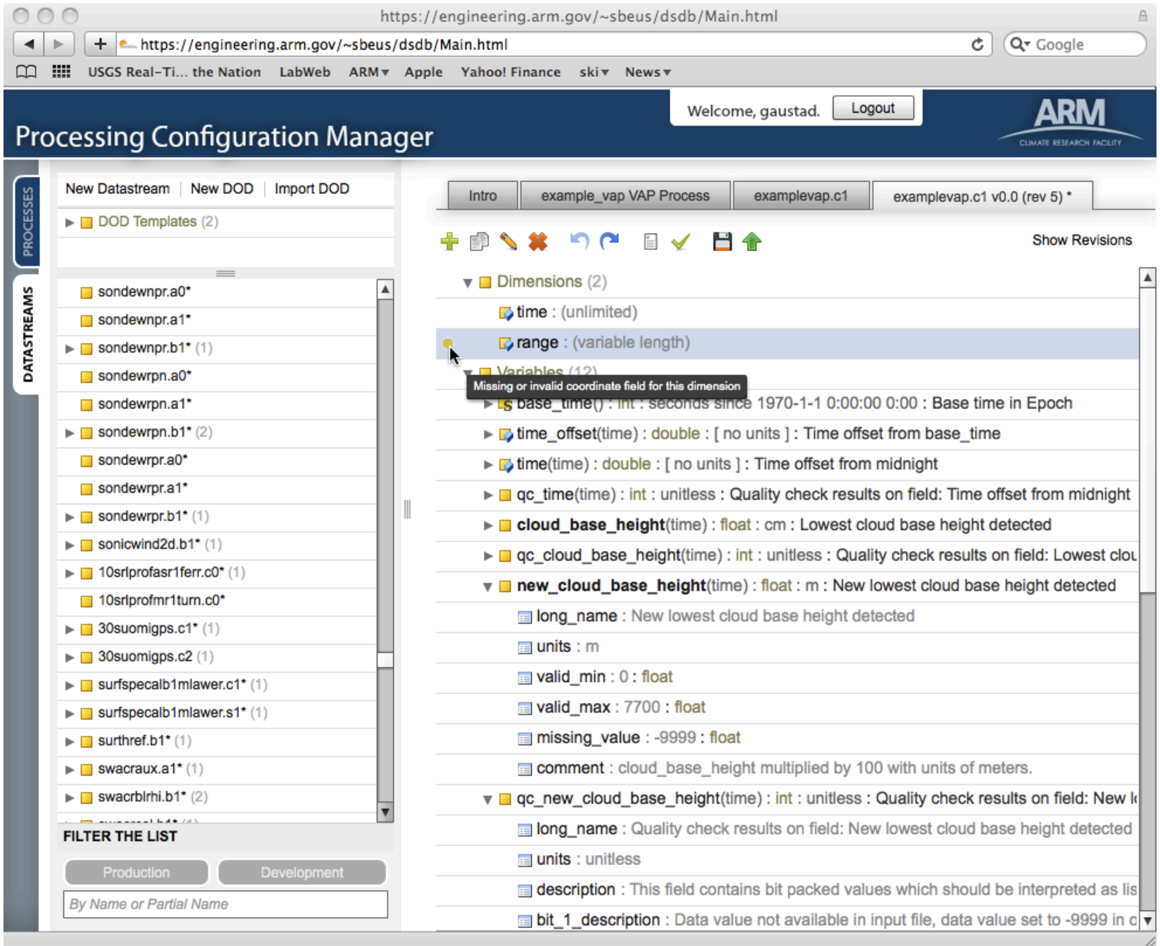


**Figure 5.10. DOD Error Messages**

It is necessary to clear all variable attributes and global attributes whose values are populated by the VAP process itself otherwise changes in these will be associated with changes in DOD version. More importantly, ISDE shared libraries will overwrite the values populated by the VAP process with the values indicated in the output DOD. As a result of a DOD that is populated by importing from an existing netCDF file, if a user does not delete the value of the 'history' global attribute, the history value from the imported file will be propagated to all output files. The error, "Attribute should not be NULL (set at run-time)", indicated by a red circle, flags such problems. Attributes that are known to be set at run time are automatically deleted by the auto-fix function executed via the check box icon. The DOD should be reviewed for non-standard attributes whose values are defined by the VAP process and manually set their values to NULL.

**CAUTION**: Periodically save the DOD as exiting the Processing Configuration Manager web page without saving will result in the loss of data.

# 6 Create C Project (isdetemplater)

A VAP's C project files include autoconf and automake files, header files, and source code. The isdetemplater function creates these files based on entries for the VAP in the DSDB and a command line template option that determines the type of project to create or the subcomponent of an existing project to recreate. The subcomponent templates result in the creation of prototypes and macros associated with either the input or output variables. Thus allowing a user to add or remove variables from either the retriever or output datastream and update the C project's file without corrupting the source code associated with the VAP's science algorithm.

The isdetemplater's command line options allow a user to specify the process for which a project should be created, add author information to code, select a template, and select the directory into which the project files will be created.

```
Usage: isdetemplater [options]
  -h, --help            show this help message and exit
  -p PROCESS, --process=PROCESS
                        Process name
  -a AUTHOR, --author=AUTHOR
                        Author name
  -n PHONE, --phone=PHONE
                        Author phone number
  -e EMAIL, --email=EMAIL
                        Author email address
  -i INPUT, --input=INPUT
                        Input file
  -t TEMPLATE, --template=TEMPLATE
                        Template to use
  -o OUTPUT, --output=OUTPUT
                        Target directory
```

## 6.1 Passthrough Template

The first time the isdetemplater is run for a new VAP, a primary template must be selected. Primary templates create all the files associated with the C project. Currently there is only one primary template available, the 'passthrough' template.

**Note**: The isdetemplater tool has limited functionality in that the only primary template that is functional is one designed to retrieve data from a single input datastream, and write out data to a single output datastream. Templates to read from multiple data input datastreams will be released in the future when the coordinate dimension functionality is incorporated into the data retrieval process.

## 6.2 Creating a C Project

a. Log onto isde.dmf.arm.gov development server and traverse to the isde home directory, $HOME/isde/isdetemplater.

b. Execute the isdetemplater function.
   $> isdetemplater -p <process name> -t passthrough -o <project directory> -a <'developer name'> -n <developer phone number> -e <your email address>
   Quotes are necessary around inputs for which there are white spaces, such as developer name and possibly phone number.

The passthrough template will create the following files in the target directory where <process_name> is the name of the process the user assigned to the VAP in step 2.3 Defining a New VAP Process. The isdetemplater will create a directory with the name indicated with the –o option, containing the following:

- AUTHORS: File with authors name
- autogen.sh – Shell script to run automake and autoconf
- ChangeLog – File to document versions
- configure.ac – Configuration file
- COPYING – Documentation file
- Makefile.am – Make file
- NEWS – Documentation file
- README – Documentation file
- src – Directory with C project source and header files.

The files in the source directory and a brief description of the purpose of each is described below.

- Makefile.am
- < process_name>_vap.c: Source code.
  - source code template (See Figure 8.1)
- < process_name>_vap.h: Main header file.
  - defines vap name, missing value, and other macros
  - defines main function prototypes
  - defines structure for accessing output datastream
- < process_name>_input_fields.h: User should not need to use the elements defined in this header file as they are are the basis for the pointers defined in the <process_name>_index_input_fields.h file. It is those pointers that the user should use to access the input data.
  - defines const char null terminated array (invarNameList) of all user defined variable names retrieved from input datastreams in order shown on Retriever Definition GUI

  - defines macros that assign an indice for each retrieved variable in order shown in Retriever Definition GUI with an assigned name of "INPUT_<user defined variable name>"
  - defines data type definitions (<input_varname>_t ) where type is type specified in Retriever Definition tool, or set to the default type value of float
  - defines an input variable structure (<vap_process>_in_t ) containing a pointer (<user defined variable name>) to each retrieved variable type.
  - defines pointer, "in" to the input variable structure.
- <process_name>_index_input_fields.h: defines references to input variables.
  - creates pointers to input data that user should use to access input data
  - pointers are defined as part of the 'in' data structure and given an name equal to the user defined name specified as the user defined name in the retriever ("in"-><user defined variable name>).
- < process_name>_index_output_fields.h: User should not need to use the elements defined in this header file as they are are the basis for the pointers defined in the <process_name>_index_ouput_fields.h file. It is those pointers that the user should use to access the output data.
  - defines a const char null terminated array (outvarNameList_<output_datastream>) containing all output variable names in order shown in datastream's DOD.
  - defines macros that assign an indice for each variable representing the order of the variables in the output datastream with assigned names of <OUTPUT_DATASTREAM>_<VARNAME>.<br>COMMENT: User probably will not need to use these as they will use the pointers defined in index_output_fields.h
  - defines data type definitions (<output_datastream>_<var_name>_t) for each variable based on output DOD type specifications.
  - defines an output variable structure (<vap_process>_out_<output_datastream>_t) containing a pointer (<var_name>) to each output variable.
  - defines an output variable structure <vap_process>_out_t that contains:
    - a pointer named <output_datastream> as a <vap_process>_out_<output_datastream>_t structure
    - an integer (<output_datastream>_dsid) reference id to the output datastream
    - a CDSGroup pointer (<output_datastream>_cds) for accessing datastream data for a single observation.
    - an integer (<output_datastream>_nvars) to access the number of variables for the output datastream
    - a CDSVar pointer to pointer to (<output_datastream>_vars) which references output variables defined the in the datastream's DOD

    - a CDSVar pointer to pointer to <output_datastream>_qc_vars) which references output QC variables defined the in the datastream's DOD
    - a CDSVar pointer to pointer to <output_datastream>_aqc_vars) which references output auxillary QC variables defined the in the datastream's DOD
    - defines pointer, "out" to the output variable structure.
  - defines a pointer, "out" to the output variable structure <vap_process>_out_t
  - defines an inline function, <vap_process>_init_output_datastreams(), that returns the datastream id of the output datastream obtained via a call to the shared library dsproc_get_output_datastream_id function.
  - defines a macro, <VAP_PROCESS>_INIT_OUTPUT_DATASTREAMS, for the <vap_process>_init_output_datastreams() function
  - defines function macro, <VAP_PROCESS>_GET_OUTPUT_DATASTREAMS that assigns address to the out-><output_datastream>_cds CDSGroup pointer via a call to dsproc_create_dataset function.
  - defines funtion macro <VAP_PROCESS>_ PROCESS_GET_OUTPUT_VARS () that assigns a value to out-><output_datastream>_nvars integer, and addresses to the out-><output_datastream>_vars, out-><output_datastream>_qc_vars , and out-><output_datastream>_acq_vars pointers via a call to dsproc_get_dataset_vars function.
  - defines function macro <VAP_PROCESS>_PROCESS_DATASET_PASS_THROUGH that copies data from input CDSGroup pointer, "in_cds" defined in <vap_process>_vap.c file) to out-><output_datastream>_cds CDSGroup structure.
  - defines function macro <VAP_PROCESS>_PROCESS_LOAD_OUTPUT_VARS that allocates data for each element of the out-><output_datastream> structure using shared library function cds_alloc_var_data, thus allocating memory for each output variable by the appropriate data type.
  - defines function macro <VAP_PROCESS>_PROCESS_STORE_DATASET that calls shared library dsproc_store_dataset function to store the <output_datastream> dataset to an output netCDF file.
- < process_name>_output_fields.h: defines references to output variables.
  - creates pointers to output data that user should use to access output data
  - creates pointers as elements of the the out-><output_datastream> structure with names equal to the variables as defined in the output datastream DOD

## 6.3 Subcomponent Templates

After the initial run of the isdetemplater with a primary template any future changes to DSDB entries defined through the PCM (i.e., datastreams, variables, and attributes) are propagated to the C project by running the isdetemplater using the appropriate subcomponent template. Each subcomponent template recreates one or more of the supporting header files.

The following subcomponent templates are available:

- autotools:  creates AUTHORS, ChangeLog, configure.ac, COPYING, Makefile.am, NEWS, README files.
- index_output_fields: creates <process_name>_index_output_fields.
- output_fields:  creates <process_name>_output fields file.
- index_input_fields:  creates <process_name>_index_input_fields file.
- input_fields:  creates <process_name>_input_fields file.
- vars:  creates the two input header files (<process_name>_index_input_fields and <process_name>_input_fields) and the two output header files (<process_name>_index_output_fields and <process_name>_output_fields).

# 7 Setup Environment, Compile and Run

Before running the new VAP, a set of environment variables, defining directories for data, configuration files, and logs associated with the VAP process must be defined. After which the automake script is run, followed by compiling and running the source code.

The typical VAP command line arguments include the following:
-s <site>
-f <facility>
-b <begin date>
-e <end date>
-a <database> (possible values are “dsdb_ref” and “devws”)
-D (to set debug)
-P (to log provenance)
-R ( reprocessing flag to allow the overwrite of previously created netCDF files).

If running from the isde.dmf.arm.gov server, the database should be specified as “dsdb_ref”. If running from a remote server on which ISDE has been installed, but not a dedicated database, access PNNL’s DSDB using –a “devws”.

The process will run for the date specified as the begin date up to the end date (i.e., NOT through the end date).

## 7.1 Define Environment Variables.

From the isde.dmf.arm.gov server perform the following (user home directory is /home/<username>)

- $> setenv DATA_HOME /data/home/<user home directory>
- $> setenv DATASTREAM_DATA /data/home/<user home directory>datastream
- $> setenv CONF_DATA /data/<user home directory>/data/conf
- $> setenv LOGS_DATA /data/<user home directory>/data/logs

## 7.2 Run Automake

From the /home/isde/isdetemplater/<target directory specified in initial call to isdetemplater function>I directory run the autogen.sh script from the command line.

$> autogen.sh

## 7.3 Compile and Run the Process

a. Change directory to the ‘src’ directory created by the autogen.sh script.
   $> cd src
b. $> Make clean
c. $> “Make”

d. $> <vap_process> -a <dsdb to access> -s <site> -f <facility> -b <YYYYMMDD> -e <YYYYMMDD> -D -R

# 8 Add Analysis to Source Code

Compiling the code in the <target directory> will produce a binary that will run "out of the box" with the capability of creating output netCDF file(s) with all passthrough variable values and completed DOD headers. The values of output variables whose values are to be calculated by VAP (i.e., the source code the user will add to the <process_name>.c file) will be populated with fill values to indicate that a value has not yet been assigned. The functions called and the tasks performed to achieve this are discussed in section 8.1 Main Process Flow. The functions used to process the data and a description of how a user can integrate new analysis into the processing portion of a VAP's execution is discussed in section 8.2 Processing Data.

## 8.1 Main Process Flow

The <process_name>_vap.c file produced by the passthrough template has all the source code unique for a particular VAP. It includes a main function that calls the ISDE DSProc's dsproc_vap_main, function, and three vap specific functions for initializing (<vap_name>_init_process), processing (<vap_name>_process_data), and finishing (<vap_name>_finish_process) the VAP process. The initialization and finishing subroutines can be used as locations to insert hooks to run new logic either before data is retrieved or after data has been processed. The dsproc_vap_main function uses the VAP's input arguments and the three VAP specific functions to manage the VAP processing. The vap specific functions passed into dsproc_vap_main are called at appropriate times by the appropriate subfunctions of dsproc_vap_main. Figure 8.1 outlines the process flow of a passthrough VAP.

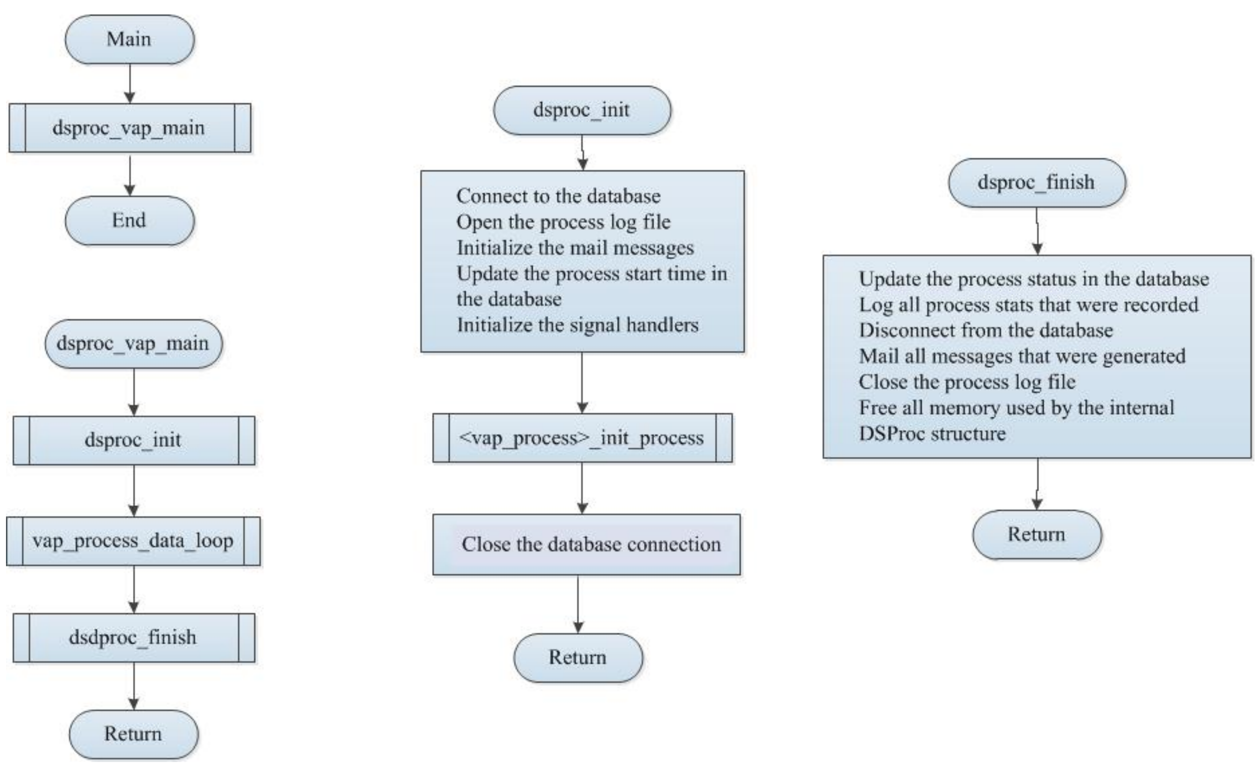


**Figure 8.1. Passthrough VAP Flow Chart**

Data is processed according to the VAP's specified processing interval. In most cases, the processing interval is a single day. The specifics of the functions involved, and the steps performed as part of the processing is discussed in the next section, 8.2 Processing Data Flow.

## 8.2 Processing Data Flow

The vap_process_data_loop function loops from the begin date to the end date as specified in the VAP's '-b' and '-e' input arguments in steps equal to the VAP's processing interval (typically a day), retrieves the input data, and calls the processing function unique to the VAP (<vap_process>_process_data).

The VAP processing function declares pointers to the output variables data types defined in the header file, sets nsamples for the output datastream, calls get dsproc_get_dataset_vars to populate an array of pointers to the output variables and output QC_companion variables that can be used assigned to data types to output variables (simplifying the readibility and use of the data in the analysis portion of the vap), and is the section of code into which a server inserts their VAP algorithm. The flow of these steps are illustrated in Figure 8.2. The code for the VAP analysis typically consists of the calculation of the new output variables. Details of the updates are outlined below:

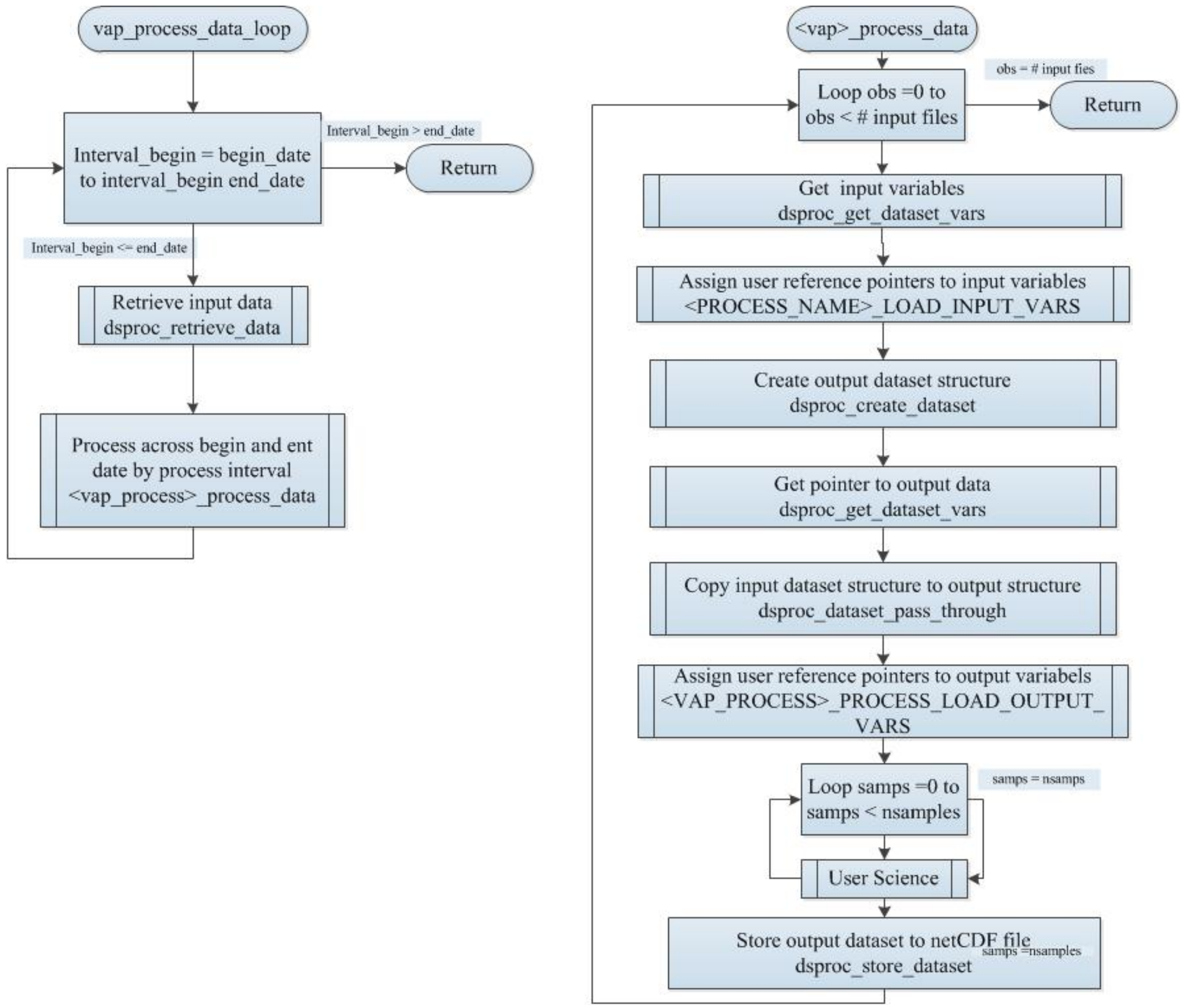


**Figure 8.2. Process Data Flow Chart**

## 8.3 Adding a Science Algorithm

The basic steps to add a science algorithm are as follows.

1. Edit <process_name>_vap.c and add science algorithm.
2. Calculate new_cloud_base_height at the cloud_base_height divided by 10 to convert it to meters.
   a. Open file $> vi <process_name>_vap.c
   b. Add in logic for calculating new value
3. Compile and Run the Process.
   c. Change directory to the ‘src’ directory created by the autogen.sh script.
      $> cd src
   d. $> Make clean

e. $> “Make”

f. $> <vap_process> -a <dsdb to access> -s <site> -f <facility> -b <YYYYMMDD> -e <YYYYMMDD> -D -R

Within the <vap_process>_vap.c file user updates will typically be able to be confined to the <vap>_process_data function. Because the pointers to all the input and output variables have been created by the isdetemplater, the user simply needs to add the analysis, and assign values to the output variables not populated by dsproc_dataset_pass_through. For the case of the example_vap, analysis will consist of assigning the new_cloud_base_height variable as cloud_base_height divided by 10 to convert it to meters. The updated source code for the example which illustrates the benefit of the previously defined pointer declarations in access variable values is shown in Figure 8.3 lines 200 to 205. A cds_print function call was also added on line 210 to illustrate its usefulness in debugging problems. The GNU Project Debugger (GDB) can also be used to debug on isde.dmf.arm.gov.

Terminal — ssh — 111×32

```
189         /* ------------------------------------------------------------ */
190         /* Start algorithm */
191         /* ------------------------------------------------------------ */
192
193         /* Loop over all samples in the dataset */
194
195         for (si = 0; si < (int)nsamples; si++) {
196
197             /*Your code here*/
198
199             /* Calculate new value as 10 * existing */
200             if(out->examplevap.cloud_base_height[si] != MISSING_VALUE) {
201               out->examplevap.new_cloud_base_height[si] = out->examplevap.cloud_base_height[si] / 10;
202             }
203             else {
204               out->examplevap.new_cloud_base_height[si] = out->examplevap.cloud_base_height[si];
205             }
206
207         } /* end samples loop */
208
209          /* Check logic by printing to debug */
210          cds_print_var(stdout, "", out->examplevap_vars[EXAMPLEVAP_NEW_CLOUD_BASE_HEIGHT],0);
211
212         /* ------------------------------------------------------------ */
213         /* End algorithm */
214         /* ------------------------------------------------------------ */
215
216
217         /*****************************************************************
218         *  Store the output dataset to disk
219         *****************************************************************/
"example_vap_vap.c" 246L, 8005C written                                          219,75        87%
```

**Figure 8.3. Adding a Science Algorithm**

## 8.4 ISDE Functions

To minimize the effort needed to add scientific algorithms to the C project created by the isdetemplater a set of Data System Process functions (libdsproc3) have been created that include wrappers to the ISDE shared libraries and specialized functions to facilitate the execution of frequently performed operations. For most VAPs the Data System Process functions should be sufficient to encode the necessary logic. However, if they are not, the user can access the ISDE foundation libraries that include functions for NetCDF (libncds3) functions for reading, writing, and performing common manipulations on netCD files, and Common Data Set (libcds3) functions that perform manipulate CDS structure elements. The primary

CDS structure is the CDSGroup that is the parent to the other CDS structures which include CDSVar, CDSAtt, CDSDim, CDSData, CDSVarGroup structures.

Pacific Northwest
NATIONAL LABORATORY

*Proudly Operated by* Battelle *Since 1965*

U.S. DEPARTMENT OF
ENERGY

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99352
1-888-375-PNNL (7665)
www.pnl.gov